

Towards Large-scale and Ultrahigh Dimensional Feature Selection via Feature Generation

Mingkui Tan

TANMINGKUI@GMAIL.COM

*School of Computer Engineering
Nanyang Technological University
642274, Singapore*

Ivor W. Tsang

IVORTSANG@NTU.EDU.SG

*School of Computer Engineering
Nanyang Technological University
639798, Singapore*

Li Wang

LIW022@UCSD.EDU

*Department of Mathematics
University of California
San Diego, USA*

Editor:

Abstract

In many real-world applications such as text mining, it is desirable to select the most relevant features or variables to improve the generalization ability, or to provide a better interpretation of the prediction models. In this paper, a novel adaptive feature scaling (AFS) scheme is proposed by introducing a feature scaling vector $\mathbf{d} \in [0, 1]^m$ to alleviate the bias problem brought by the scaling bias of the diverse features. By reformulating the resultant AFS model to semi-infinite programming problem, a novel feature generating method is presented to identify the most relevant features for classification problems. In contrast to the traditional feature selection methods, the new formulation has the advantage of solving extremely high-dimensional and large-scale problems. With an exact solution to the worst-case analysis in the identification of relevant features, the proposed feature generating scheme converges globally. More importantly, the proposed scheme facilitates the group selection with or without special structures. Comprehensive experiments on a wide range of synthetic and real-world datasets demonstrate that the proposed method achieves better or competitive performance compared with the existing methods on (group) feature selection in terms of generalization performance and training efficiency. The C++ and MATLAB implementations of our algorithm can be available at <http://c2inet.sce.ntu.edu.sg/Mingkui/robust-FGM.rar>.

Keywords: Feature selection, feature scaling bias, multiple kernel learning

1. Introduction

During the past decades, feature selection problem has been one of the most important tasks in the realm of machine learning. There are several reasons accounting for this. Firstly, many datasets such as text, image and Microarray data are represented in very high dimensional vectors, which not only incur huge computational expense during the

learning process, but also deteriorate the learning ability, which is also known as “the curse of dimensionality” (Duda et al., 2000.; Guyon and Elisseeff, 2003; Zhang and Lee, 2006; Dasgupta et al., 2007; Blum et al., 2007). Secondly, many features in high dimensional vectors are usually non-informative or noisy to the output labels and hence may seriously confuse the prediction models. Thus, selecting the most informative features can vastly improve the generalization performance (Ng, 1998). Thirdly, a sparse classifier leads to a simplified decision rule for faster prediction on large-scale problems. Finally, in some applications like Microarray data analysis (Guyon and Elisseeff, 2003), a small subset of input features is expected to interpret the results for further analysis.

Feature selection methods are mainly categorized into two groups: filter methods and wrapper methods (Kohavi and John, 1997; Ng, 1998). Filter methods refer to selecting the informative features according to their discriminative power or correlation criterion without considering any knowledge of the predictive model; while wrapper methods select the discriminative features mainly based on the inductive learning rules. The advantages of filter methods lie in their low computational requirements and hence the high efficiency in dealing with large-scale problems. The drawback however is that it may not identify the optimal feature subset suitable for the predictive model of interest. Signal-to-noise ratio method (Golub et al., 1999) and spectral feature filtering (Zhao and Liu, 2007) are typical examples of filter methods. Compared with filter methods, wrapper methods are expected to be of higher performance on accuracy. However, owing to the expensive computational cost, one critical bottleneck for wrapper methods is their scalability to large scale and very high dimensional problems. Thereafter, the recent focus is mainly on how to improve the efficiency of wrapper methods.

Regarding wrapper methods, recently, numerous feature selection methods based on support vector machine (SVM) have been proposed for classification problems (Blum and Langley, 1997; Guyon and Elisseeff, 2003). One popular SVM based feature selection is the Recursive Feature Elimination (RFE) scheme proposed by Guyon et al. (2002). SVM-RFE obtains nested subsets of input features and has shown good performance on gene selection task in Microarray data analysis (Guyon et al., 2002). However, as indicated by Xu et al. (2009a), such “monotonic” nested feature selection scheme is suboptimal in identifying the most informative subset of input features. Herein, the “monotonic” property refers to the problem that, if an informative feature is wrongly eliminated from a subset \mathcal{S} , it will not be in the nested subsets of \mathcal{S} (Xu et al., 2009a). Rather than using the recursive elimination scheme, Xu *et al.* proposed a non-monotonic feature selection method, namely NMMKL, which approximates the combinatorial optimization problem of feature selection. However, their method solves a quadratically constrained quadratic programming (QCQP) problem with m quadratic constraints, where m denotes the number of input features. However, NMMKL is computationally infeasible for high dimensional problems. Besides these schemes, many researchers (Bradley and Mangasarian, 1998; Zhu et al., 2003; Fung and Mangasarian, 2004) proposed ℓ_1 -norm Sparse SVM (SSVM), which uses $\|\mathbf{w}\|_1$ as the regularizer, where \mathbf{w} is the weight vector of the decision function. The resultant problem is convex, and can be solved optimally by Newton method (Fung and Mangasarian, 2004) or coordinate descent methods (Yuan et al., 2010, 2011). Recently, Chan et al. (2007) proposed another two convex relaxations to the ℓ_0 -norm SSVM, namely QCQP-SSVM and SDP-SSVM, which can be solved by convex QCQP and semi-definite programming (SDP),

respectively. Though both the relaxed optimization problems are convex, they are computationally expensive, especially for high dimensional problems.

Feature selection relates to the Multiple Kernel Learning (MKL) problem. During the last decades, MKL has been proposed to learn good kernels for supervised problems (Lanckriet et al., 2004; Bach et al., 2004; Sonnenburg et al., 2006; Rakotomamonjy et al., 2008; Xu et al., 2010; Kloft et al., 2011). By learning the optimal linear combination of base kernels, MKL can be considered as a special case of feature selection methods (Sonnenburg et al., 2006). Lanckriet et al. (2004) first proposed the use of QCQP in MKL. Bach et al. (2004) then showed that an approximate solution can be efficiently obtained by Sequential Minimization Optimization (SMO). Recently, Sonnenburg et al. (2006) proposed a semi-infinite linear programming formulation which allows MKL to be iteratively solved with standard SVM solver and linear programming. More recently, Xu et al. (2009b) proposed the use of the extended level method to further improve the convergence of MKL. However, before applying MKL, one needs to predefine a set of base kernels which are usually constructed by using parts of the input features. However, it is non-trivial to determine such subsets of input features from the exponentially combinatorial space. A popular strategy is to randomly select part of the input features to construct the base kernels (Bach et al., 2004; Sonnenburg et al., 2006), which is obviously suboptimal to feature selection especially for high dimensional problems. Another critical issue of MKL learning is the high computation cost brought by the extremely large number of features or kernels (Bach, 2009), which brings a great challenge in optimization.

Recently, Tan et al. (2010) introduced a novel ℓ_0 -norm SSVM model to select features by introducing a feature indicator vector $\mathbf{d} \in \{0, 1\}^m$ to the standard SVM. The resultant ℓ_0 -norm sparse SVM is transformed to a mixed integer programming (MIP) problem. After that, a feature generating scheme (FGM) is proposed to solve a convex relaxation of the MIP problem. Because of its nice optimization framework, the feature generating scheme has brought benefits to other applications. For example, the proposed feature generating scheme has been successfully applied to image retrievals (Rastegari et al., 2011), multi-label predictions (Gu et al., 2011a) and graph-based feature selection methods (Gu et al., 2011b) by several researchers. In view of the remained issues in (Tan et al., 2010), this paper makes the following extensions and improvements.

- We illustrate the feature generating strategy from the perspective of adaptive feature scaling (AFS) schemes by imposing an ℓ_1 constraint on a feature scaling vector $\mathbf{d} \in [0, 1]^m$. Under the AFS scheme, as will be stated in Theorem 1, the resultant problem can be reformulated as a semi-infinite convex programming problem without any convex relaxations. Furthermore, with the separation of the complexity control and sparsity control of the proposed AFS scheme, the bias problem can be alleviated.
- We extend in this paper the feature generating strategy to group selections with or without special structures via an adaptive group scaling (AGS) scheme. We also extend FGM to learn nonlinear features by either kernelization or using the explicit feature mappings. Although general nonlinear feature selection is still very challenging, we show that under some conditions, FGM can obtain the optimal feature subset for some special kernels. Furthermore, we also incorporate the feature generating scheme to logistic regression models.

- Thirdly, we observe that the dual variables for the square hinge loss and logistic loss can be easily recovered by the training loss. Instead of using SimpleMKL algorithm to solve the MKL subproblem as in (Tan et al., 2010), in this paper, we propose to solve MKL subproblem in its primal via an accelerated proximal gradient descent method. This primal optimization strategy tremendously improves the convergence of FGM. After that, the recovered dual variables can be used for the worst-case analysis. From our experiments, the new implementation can gain up to 1000 times faster than that with SimpleMKL solver (Rakotomamonjy et al., 2008).
- Fourthly, efficient cache techniques are developed to significantly alleviate the memory requirement of feature selections for extremely high dimensional datasets and the learning problems with explicit nonlinear feature mappings.
- Last but not least, comprehensive experiments on (group) feature selections verify that the proposed method achieves better or competitive prediction accuracy when compared with the related state-of-the-art feature selection methods. Particularly, the proposed method shows great scalability on large-scale and extremely high dimensional datasets.

The rest of this paper is organized as follows. Section 2 describes the adaptive feature scaling (AFS) and adaptive group scaling (AGS) based feature selections, which can be unified as a semi-infinite programming problem. The feature generating scheme as well as its convergence behavior will be described in Section 3. The worst-case analysis and the subproblem learning are presented in Section 4 and Section 5, respectively. The experiments on linear feature selection are presented in Section 6; while the experiments on group and nonlinear feature selection are presented in 7. The last section will give some discussions and conclusive remarks.

2. Feature Selection via Adaptive Feature Scaling

In the sequel, we denote the transpose of vector/matrix by the superscript $'$ and a vector with all entries equal to one as $\mathbf{1} \in \mathbb{R}^n$, and a zero vector $\mathbf{0} \in \mathbb{R}^n$. In addition, we denote a dataset by $\mathbf{X} = [\mathbf{f}_1, \dots, \mathbf{f}_m] = [\mathbf{x}'_1, \dots, \mathbf{x}'_n]'$, where $\mathbf{f}_i \in \mathbb{R}^n$ denotes the i th feature vector and $\mathbf{x}_j \in \mathbb{R}^m$ represents the j th instance. In addition, we use $|\mathcal{G}|$ to denote the length of an index set \mathcal{G} and $|v|$ to denote the absolute value of a real number v . In addition, we denote $\mathbf{v} \succeq \alpha$ if $v_i \geq \alpha, \forall i$ and $\mathbf{v} \preceq \alpha$ if $v_i \leq \alpha, \forall i$. We also denote $\|\mathbf{v}\|_p$ as the ℓ_p -norm of a vector, $\|\mathbf{v}\|$ as the ℓ_2 -norm of a vector, $\|\mathbf{v}\|_1^2 = (\|\mathbf{v}\|_1)^2$ as the ℓ_1^2 regularizer and $\|\mathbf{A}\|_{21} = \sum_{i=1}^p \|\mathbf{a}_i\|$ with $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_p]$. Finally, $\mathbf{A} \odot \mathbf{B}$ represents the element-wise product between two matrices \mathbf{A} and \mathbf{B} .

2.1 Feature Selection via ℓ_1 -Sparse Support Vector Machines

Given a set of labeled patterns $\{\mathbf{x}_i, y_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^m$ is the input and $y_i \in \{\pm 1\}$ is the output label, we can learn a linear decision hyperplane $f(x) = \mathbf{w}'\mathbf{x}$ that minimizes the

following structural risk functional:

$$\min_{\mathbf{w}} \Omega(\mathbf{w}) + C \sum_{i=1}^n l(-y_i \mathbf{w}' \mathbf{x}_i), \quad (1)$$

where $\mathbf{w} \in \mathbb{R}^m$ is the weight vector of the decision hyperplane, $\Omega(\mathbf{w})$ is the regularizer that defines the characteristic (e.g. sparsity) of \mathbf{w} , $l(\cdot)$ is a convex loss function¹, and $C > 0$ is a regularization parameter that trades off the model complexity and the fitness of the decision hyperplane. For standard SVM, $\Omega(\mathbf{w})$ is set to $\frac{1}{2} \|\mathbf{w}\|_2^2$, a non-sparse regularizer and the learned \mathbf{w} is usually non-sparse. To obtain a sparse decision rule for SVM, one of the most widely used approaches is to introduce ℓ_1 -regularizer on the loss function (Bradley and Mangasarian, 1998; Zhu et al., 2003; Fung and Mangasarian, 2004), which will produce the following the ℓ_1 -norm SVM problem.

$$\min_{\mathbf{w}} \|\mathbf{w}\|_1 + C \sum_{i=1}^n l(-y_i \mathbf{w}' \mathbf{x}_i), \quad (2)$$

Although the level of sparsity in the decision function of (2) can be controlled by varying C , due to the regularization term $\|\mathbf{w}\|_1$, the bias problem inevitably exists in ℓ_1 -regularized optimization problems (Figueiredo et al., 2007). Specifically, for a feature j , its weight w_j would be very large if it is very informative for the classification and important to reduce the empirical loss. However, its value will be suppressed due to the minimization of $\|\mathbf{w}\|_1$ in (2). Then the scaling bias happens. In addition, due to the scale variation of \mathbf{w} , it is hard to control its sparsity, meanwhile, to regulate the decision function. Finally, the ℓ_1 -regularization can be inefficient when solving large-scale and very high dimensional problems, especially on those datasets that are too large to be loaded into the memory storage. To address these issues, in the following, we will propose to separate the regularization term and the sparsity control of the decision function via a novel feature scaling scheme (AFS).

2.2 Adaptive Feature Scaling

Adaptive feature scaling (AFS) has been widely used for feature selections (Weston et al., 2000; Chapelle et al., 2002; Grandvalet and Canu, 2002; Rakotomamonjy, 2003; Varma and Babu, 2009). In AFS scheme, a feature scaling vector $\boldsymbol{\delta} \succeq 0$ is introduced to the features and then the feature selection problem can be treated as a model selection problem (Weston et al., 2000). To solve it, a general scheme is by minimizing a bound on the estimate of the generalization error iteratively (Weston et al., 2000; Rakotomamonjy, 2003) or a regularized joint problem with respect to both the weight vector \mathbf{w} and the hyperparameters $\boldsymbol{\delta}$ (Grandvalet and Canu, 2002; Varma and Babu, 2009). By incorporating the inductive learning algorithms (for example, SVM), these methods have shown good performance on small problems. However, several issues still remain for the above AFS strategies. At first, because $\boldsymbol{\delta}$ is imposed on the features, the whole optimization problem is non-convex. Hence, these methods can only guarantee the local optimal solutions even for linear problems. Moreover, because the update of $\boldsymbol{\delta}$ involves all input features and the

1. In the context of SVMs, the maximum margin decision hyperplane is usually learned based on either the hinge loss or the square hinge loss.

model selection needs many machine re-trainings, all these methods cannot handle large-scale and very high-dimensional problems. Finally, it is also non-trivial to explicitly control the number of selected features via the iterative minimization process in the AFS.

Notice that, in many real-world feature selection applications, it is more practical to select a desired number of features while attaining acceptable generalization performance, which can be considered as the priors for learning problems. Although such prior is not necessarily very precise, it can ease the problem of model selections. What's more, for many applications, we indeed have some prior knowledge about the number of features to be selected. For example, in the Microarray analysis, due to expensive bio-diagnosis and limited resources, biologists prefer to select less than 100 genes from hundreds of thousands of genes (Guyon et al., 2002; Golub et al., 1999).

By taking these facts into consideration, we introduce a new adaptive feature scaling scheme as follows. At first, similar in the traditional AFS scheme, we also introduce a feature scaling vector $\mathbf{d} \in \mathbb{R}^m$ to measure the importance of features and bound it such that $0 \preceq \mathbf{d} \preceq 1$. That is, every d_i is in the range of $[0, 1]$. Different from the traditional AFS scheme, we use the following constraint to explicitly control the sparsity of the decision hyperplane.

$$\sum_{j=1}^m d_j = \|\mathbf{d}\|_1 \leq B, \quad d_j \in [0, 1], \quad j = 1, \dots, m, \quad (3)$$

where we select the j th feature if and only if $d_j > 0$ and drop the j th feature if $d_j = 0$. Let $\mathcal{D} = \{\mathbf{d} \in \mathbb{R}^m \mid \sum_{j=1}^m d_j \leq B, d_j \in [0, 1], j = 1, \dots, m\}$ be the domain of \mathbf{d} , and B controls the sparsity of \mathbf{d} . Here B can be considered as the prior knowledge about the number of features that we prefer to select. In addition, to enforce the non-negative feature scaling, rather than imposing \mathbf{d} to the features, we impose $\sqrt{\mathbf{d}} = [\sqrt{d_1}, \dots, \sqrt{d_m}]'$ to the features. Finally, to avoid the scaling bias brought by the data scales, we can optionally introduce an additional vector $\boldsymbol{\lambda} = [\frac{1}{\|\mathbf{f}_1\|}, \dots, \frac{1}{\|\mathbf{f}_m\|}]' \in \mathbb{R}^m$ to the features², and then we have:

$$\mathbf{w}'(\boldsymbol{\lambda} \odot \sqrt{\mathbf{d}} \odot \mathbf{x}) = (\mathbf{w} \odot \boldsymbol{\lambda} \odot \sqrt{\mathbf{d}})' \mathbf{x}. \quad (4)$$

Notice that $\boldsymbol{\lambda}$ can be also some priors. For general problems, we can do preprocessing on \mathbf{X} by $\boldsymbol{\lambda} \odot \mathbf{x}$, and then the term $\boldsymbol{\lambda}$ can be absorbed. However, in some cases where m is extremely large, the preprocessing will be very costly and it is desirable to do such processing dynamically. Hence we keep $\boldsymbol{\lambda}$ for completeness. Herein, we focus on the squared hinge loss³, and the proposed sparse formulation can be written as:

$$\begin{aligned} \min_{\mathbf{d} \in \mathcal{D}, \mathbf{w}, \boldsymbol{\xi}} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{C}{2} \sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & y_i \mathbf{w}'(\mathbf{x}_i \odot \boldsymbol{\lambda} \odot \sqrt{\mathbf{d}}) \geq 1 - \xi_i, \quad i = 1, \dots, n, \end{aligned} \quad (5)$$

where constant C is a regularization parameter that makes a trade-off between the model complexity and the fitness of the decision hyperplane; while B controls the number of selected features.

2. It is possible that some $\|\mathbf{f}_i\| = 0$ on sparse datasets. We directly remove these features or define $\frac{0}{0} = 0$.

If the data \mathbf{X} is well scaled, we can simply set $\boldsymbol{\lambda} = [1, \dots, 1]'$.

3. Obviously, we can also apply the same scheme to the hinge loss and the logistic loss.

There are several advantages of imposing ℓ_1 -regularization on \mathbf{d} rather than on \mathbf{w} in (6). At first, in AFS, the ℓ_2 -regularization term on \mathbf{w} to control the complexity and the sparsity term on \mathbf{d} are now separated in (6). Since $0 \preceq \mathbf{d} \preceq \mathbf{1}$, in spite of the variation of the entries in \mathbf{w} , $\|\mathbf{d}\|_1$ keeps relatively stable and reduces the negative influence brought by the scale variation of w_i . In addition, the parameter B represents the minimal number of features that we desire to select. Obviously, it is easier to interpret the parameter B than the parameter C in feature selection tasks. Recall that the ℓ_1 -regularized problem in (2) can be written as a constrained optimization problem $\min_{\mathbf{w}} \sum_{i=1}^n l(-y_i \mathbf{w}' \mathbf{x}_i)$ under the sparse constraint $\|\mathbf{w}\|_1 \leq B$. Still, it is non-trivial to control the sparsity via $\|\mathbf{w}\|_1 \leq B$ because of the variation of values in \mathbf{w} . On the other hand, introducing the scalar B in (5) will bring convenience for controlling the number of selected features, which will be discussed in details in the experiments. Last but not least, we can transform the adaptive scaling problem into a convex optimization problem, which will benefit for both optimization and model selection.

Notice that problem (5) can be rewritten as:

$$\begin{aligned} \min_{\mathbf{d} \in \mathcal{D}} \min_{\mathbf{w}, \boldsymbol{\xi}} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{C}{2} \sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & y_i \mathbf{w}' (\mathbf{x}_i \odot \boldsymbol{\lambda} \odot \sqrt{\mathbf{d}}) \geq 1 - \xi_i, \quad i = 1, \dots, n, \end{aligned} \quad (6)$$

where the inner minimization problem regarding \mathbf{w} and $\boldsymbol{\xi}$ for a fixed \mathbf{d} is a standard SVM problem.

Proposition 1 *For a fixed \mathbf{d} , the inner minimization problem of (6) can be solved by its dual, so problem (6) is equivalent to the following problem:*

$$\min_{\mathbf{d} \in \mathcal{D}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} -\frac{1}{2} \left\| \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i \odot \boldsymbol{\lambda} \odot \sqrt{\mathbf{d}}) \right\|^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha}, \quad (7)$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the vector of dual variables for the inequality constraints of the inner minimization problem in (6), and $\mathcal{A} = \{\boldsymbol{\alpha} | \alpha_i \geq 0, i = 1, \dots, n\}$ is the domain of $\boldsymbol{\alpha}$.

Proof The proof can be completed by fixing \mathbf{d} and introducing the SVM dual problem of the inner minimization problem regarding \mathbf{w} and $\boldsymbol{\xi}$. ■

Notice that \mathcal{A} is a convex set but not compact. Practically, $\boldsymbol{\alpha}$ is usually bounded, so we may bring in a big number $l > 0$ such that $\boldsymbol{\alpha}^* \in \mathcal{A} = \{\boldsymbol{\alpha} | 0 \leq \alpha_i \leq l, i = 1, \dots, n\}$, then both \mathbf{d} and $\boldsymbol{\alpha}$ are in compact domain. Then let $\boldsymbol{\omega} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$, we have $\|\sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i \odot \boldsymbol{\lambda} \odot \sqrt{\mathbf{d}})\|^2 = \sum_{j=1}^m d_j \lambda_j^2 \omega_j^2$.

Theorem 1 *With the definition of \mathcal{A} and \mathcal{D} above, by interchanging the order of $\min_{\mathbf{d} \in \mathcal{D}}$ and $\max_{\boldsymbol{\alpha} \in \mathcal{A}}$ in (7), the following equality holds,*

$$\min_{\mathbf{d} \in \mathcal{D}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} -\frac{1}{2} \sum_{j=1}^m d_j \lambda_j^2 \omega_j^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha} = \max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\mathbf{d} \in \mathcal{D}} -\frac{1}{2} \sum_{j=1}^m d_j \lambda_j^2 \omega_j^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha} \quad (8)$$

Proof Define $f(\boldsymbol{\alpha}, \mathbf{d}) = -\frac{1}{2} \sum_{j=1}^m d_j \lambda_j^2 \omega_j^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha}$. Obviously, $f(\boldsymbol{\alpha}, \mathbf{d})$ is linear in \mathbf{d} and concave in $\boldsymbol{\alpha}$. Then the above equality holds according to the minimax saddle-point theorem (Kim and Boyd, 2008). \blacksquare

Furthermore, the equality of (8) states that, rather than directly solving (7), which is of NP-hardness, we can equivalently address it by solving the following maximin problem.

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\mathbf{d} \in \mathcal{D}} -\frac{1}{2} \sum_{j=1}^m d_j \lambda_j^2 \omega_j^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha}. \quad (9)$$

Remark 1 Recall that in FGM (Tan et al. (2010)), the feature selection vector \mathbf{d} is a zero-one vector, i.e., $\mathbf{d} \in \{0, 1\}^m$. Then the resultant problem is a MIP problem where we only have

$$\min_{\mathbf{d} \in \{0, 1\}^m} \max_{\boldsymbol{\alpha} \in \mathcal{A}} f(\boldsymbol{\alpha}, \mathbf{d}) \geq \max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\mathbf{d} \in \{0, 1\}^m} f(\boldsymbol{\alpha}, \mathbf{d}).$$

In other words, the latter problem is only the convex relaxation of the former problem. And it is very hard to measure the gap between them.

2.3 Adaptive Group Scaling

The AFS can be easily extended to group selections. Recently, feature selection with group priors has attracted increasing attentions in real applications. With group information of features provided, the features in one group can be selected only if this group is selected. Let $\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_p\}$ be a set of predefined groups with $|\mathcal{G}| = p$, where $\mathcal{G}_j \subset \{1, \dots, m\}, j = 1, \dots, p$, denotes the index set of feature support. For simplicity, we first assume that there is no overlapping features between groups, namely, $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \forall i \neq j$. In order to control the sparsity regarding the groups, we again introduce a group scaling vector $\hat{\mathbf{d}} = [\hat{d}_1, \dots, \hat{d}_p]' \in \hat{\mathcal{D}}$ to control the sparsity of groups. If there is no overlapping among groups, similar in the linear feature selection, we have:

$$f(\mathbf{w}) = \mathbf{w}' \mathbf{x} = \sum_{j=1}^p \sqrt{\hat{d}_j} \mathbf{w}'_{\mathcal{G}_j} \mathbf{x}_{\mathcal{G}_j}, \quad (10)$$

where $\hat{\mathcal{D}} = \{\hat{\mathbf{d}} \in \mathbb{R}^p \mid \sum_{j=1}^p \hat{d}_j \leq B, \hat{d}_j \in [0, 1], j = 1, \dots, p\}$ is the domain of $\hat{\mathbf{d}}$ and the scalar B controls the sparsity of $\hat{\mathbf{d}}$. To avoid the bias brought by the group size and data scales, we can (optionally) introduce an additional vector λ_j to each group \mathcal{G}_j , and we will have:

$$f(\mathbf{w}) = \mathbf{w}' \mathbf{x} = \sum_{j=1}^p \lambda_j \sqrt{\hat{d}_j} \mathbf{w}'_{\mathcal{G}_j} \mathbf{x}_{\mathcal{G}_j}, \quad (11)$$

where λ_j can be prior parameters to the groups (Martins et al., 2010). Although it can be absorbed by preprocessing, we keep it for the completeness of the presentation. Then the

group selection problem for square hinge loss can be formulated as follows:

$$\begin{aligned} \min_{\hat{\mathbf{d}} \in \hat{\mathcal{D}}, \mathbf{w}, \xi_i} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{C}{2} \sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & y_i \sum_{j=1}^p \lambda_j \sqrt{\hat{d}_j} \mathbf{w}'_{\mathcal{G}_j} \mathbf{x}_{i\mathcal{G}_j} \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, n. \end{aligned} \quad (12)$$

With the similar deductions in Section 2, we obtain the following minimax problem:

$$\min_{\hat{\mathbf{d}} \in \hat{\mathcal{D}}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} -\frac{1}{2} \sum_{j=1}^p \lambda_j^2 \hat{d}_j \left\| \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{i\mathcal{G}_j} \right\|^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha}, \quad (13)$$

Furthermore, let $\boldsymbol{\omega}_{\mathcal{G}_j} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}$, similarly in the linear feature selection, by interchanging the order of min and max operators, we can have the following equivalence:

$$\min_{\hat{\mathbf{d}} \in \hat{\mathcal{D}}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} -\frac{1}{2} \sum_{j=1}^p \lambda_j^2 \hat{d}_j \|\boldsymbol{\omega}_{\mathcal{G}_j}\|^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha} = \max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\hat{\mathbf{d}} \in \hat{\mathcal{D}}} -\frac{1}{2} \sum_{j=1}^p \lambda_j^2 \hat{d}_j \|\boldsymbol{\omega}_{\mathcal{G}_j}\|^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha} \quad (14)$$

Recall that (14) and (8) have the similar formulations, and if $|\mathcal{G}_j| = 1, j = 1, \dots, p$, namely, the size for each group is 1, (14) reduces to problem (8). Therefore, for simplicity, we can solve a unified problem as follows,

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\mathbf{d} \in \mathcal{D}} -\frac{1}{2} \sum_{j=1}^p d_j \lambda_j^2 \boldsymbol{\omega}_{\mathcal{G}_j}^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha}, \quad (15)$$

where $\mathcal{D} = \{\mathbf{d} \in \mathbb{R}^p \mid \sum_{j=1}^p d_j \leq B, d_j \in [0, 1], j = 1, \dots, p\}$ is the domain of \mathbf{d} , where $p = m$ if $|\mathcal{G}_j| = 1, \forall j = 1, \dots, p$. Furthermore, by defining $f(\boldsymbol{\alpha}, \mathbf{d}) = -\frac{1}{2} \sum_{j=1}^p \lambda_j^2 d_j \|\boldsymbol{\omega}_{\mathcal{G}_j}\|^2 - \frac{1}{2C} \boldsymbol{\alpha}' \boldsymbol{\alpha}$ and bringing in an additional variable $\theta \in \mathbb{R}$, the above maximin problem can be transformed into the following convex semi-infinite QCQP problem (Pee and Royset, 2010),

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}, \theta} -\theta \quad : \quad \theta \geq -f(\boldsymbol{\alpha}, \mathbf{d}), \quad \forall \mathbf{d} \in \mathcal{D}. \quad (16)$$

2.4 Adaptive Groups Scaling with Special Structures

The adaptive group scaling (AGS) scheme described above can be easily extended for groups with special structures, such as overlapping groups or tree-structured groups. A simple way to handle the overlapping groups is to augment the dataset $\mathbf{X} = [\mathbf{f}_1, \dots, \mathbf{f}_m]$ with duplicated features, and to explicitly make the overlapping groups into non-overlapping groups. For example, suppose we have a dataset $\mathbf{X} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3]$ with group prior $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2\}$ with $\mathcal{G}_1 = \{1, 2\}$ and $\mathcal{G}_2 = \{2, 3\}$. As a results, we can obtain the dataset as $\mathbf{X}_e = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_2, \mathbf{f}_3]$. The feature augmentation idea can be also extended to even more complex structures, such as tree structures or graph structures (Bach, 2009). Here we only discuss the tree-structured groups, which can be formally defined as follows (Jenatton et al., 2011; Kim and Xing, 2010, 2012).

Definition 1 (*Tree-structured set of groups.*) A super set of groups $\mathcal{G} \triangleq \{\mathcal{G}_h\}_{\mathcal{G}_h \in \mathcal{G}}$ with $|\mathcal{G}| = p$ is said to be tree-structured in $\{1, \dots, m\}$, if $\cup \mathcal{G}_h = \{1, \dots, m\}$ and if for all $\mathcal{G}_g, \mathcal{G}_h \in \mathcal{G}$, $(\mathcal{G}_g \cap \mathcal{G}_h \neq \emptyset) \Rightarrow (\mathcal{G}_g \subseteq \mathcal{G}_h \text{ or } \mathcal{G}_h \subseteq \mathcal{G}_g)$. For such a set of groups, there exists a (non-unique) total order relation \preceq such that:

$$\mathcal{G}_g \preceq \mathcal{G}_h \Rightarrow \{\mathcal{G}_g \subseteq \mathcal{G}_h \text{ or } \mathcal{G}_g \cap \mathcal{G}_h = \emptyset\}.$$

Similar to the overlapping case, we can augment the dataset with all possible groups along the tree structures, resulting in an augmented data set $\mathbf{X}_{\mathcal{G}} = [\mathbf{X}_{\mathcal{G}_1}, \dots, \mathbf{X}_{\mathcal{G}_p}]$, where $\mathbf{X}_{\mathcal{G}_i}$ presents the data columns selected by \mathcal{G}_i and p denotes the number of all possible groups. Although this idea is simple, it may bring great challenges for optimization when there are huge number of overlapping groups. Specifically, the number of groups p can be exponential in m for graph based group structures (Bach, 2009). Fortunately, as only a small number of groups will be selected as the informative groups, the computational issue mentioned above can be easily solved with the following proposed feature generating scheme and will be detailed in Section 4.3.

3. Optimization by Feature Generation

From the above deduction, the feature selection problem either with or without group priors can be unified as a QCQP optimization problem as in equation (16). However, it is very hard to solve as there are infinite number of quadratic inequality constraints in (16), which is in the form of the semi-infinite programming (SIP) problem (Kelley, 1960). Regarding such SIP problems, the cutting plane algorithm is an efficient optimization scheme to solve it. Moreover, as we only need to select a small number of features or feature groups, which indicates that only a small number of the constraints in the QCQP problem are active at the optimality. Hence including only a subset of these constraints usually leads to a very tight approximation of the original optimization problem. In this sense, the feature selection problem can be solved efficiently by the cutting plane algorithm (Kelley, 1960; Mutapcic and Boyd, 2009).

Based on the detailed discussion in (Kelley, 1960; Mutapcic and Boyd, 2009), the cutting plane algorithm for solving the SIP problem (16) can be presented in Algorithm 1.

Algorithm 1 The cutting plane algorithm for FGM.

- 1: Initialize $\alpha = \mathbf{1}$ and the constraint subset $\mathcal{C} = \emptyset$. Let iteration index $T = 1$.
 - 2: **Feature Generation Stage:** Find the most violated \mathbf{d}_T via the worst-case analysis, and set $\mathcal{C} = \mathcal{C} \cup \{\mathbf{d}_T\}$.
 - 3: **Subproblem Learning Stage:** Solve the subproblem (17) to obtain an updated α .
 - 4: Let $T = T + 1$. Repeat step 2-3 until convergence.
-

The basic idea is that, starting from an initial solution α , we iteratively generate an active constraint via the worst-case analysis procedure (Kelley, 1960; Mutapcic and Boyd, 2009) and add it into an active constraint set \mathcal{C} , which is initialized as empty set \emptyset . In addition, at the initialization stage, we can initialize the vector of dual variables α to $\mathbf{1}$, which is the empirical loss of every instance without selecting any features. Based on a given α , at the T th iteration, we can find the most-violated (or active constraint) \mathbf{d}_T via

the worst-case analysis procedure and then include it into the active constraint set \mathcal{C} by $\mathcal{C} \cup \{\mathbf{d}_T\}$. Finally, once we obtain a new active constraint set \mathcal{C} , we will need to solve a reduced QCQP problem defined by \mathcal{C} to update the dual variable $\boldsymbol{\alpha}$. Specifically, we need to solve the following standard problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in \mathcal{A}, \theta \in \mathbb{R}} \quad & \theta \\ \text{s.t.} \quad & f(\boldsymbol{\alpha}, \mathbf{d}) - \theta \leq 0, \quad \forall \mathbf{d} \in \mathcal{C}. \end{aligned} \quad (17)$$

As $|\mathcal{C}|$ is very small, one can efficiently solve (17) and obtain a new $\boldsymbol{\alpha}$ to update the active constraint set \mathcal{C} via the worst-case analysis. The whole procedure will iterate until the stopping condition is achieved. As from (Mutapcic and Boyd, 2009), such cutting plane algorithm can converge to a robust optimal point within tens of iterations with the exact worst-case analysis. Notice that in the worst-case analysis for the T th iteration, as will be discussed in Section 4, \mathbf{d} includes a set of B features or B feature groups. With this reason, hereafter, we term the worst-case analysis step as the **Feature Generation Stage** and Algorithm 1 as **Feature Generating Machine (FGM)**, respectively. Correspondingly, we term the step of solving the subproblem (17) as the **Subproblem Learning Stage**.

Before the detailed discussion of the worst-case analysis and the solution to the subproblem (17), we first consider the convergence properties of FGM given that we can exactly solve the above two problems.

Specifically, let $\mathcal{A} \times \mathcal{D}$ be the constraint domain for problem (16). In FGM, we iteratively find and add the most violated constraint to the set \mathcal{C} , which is a subset of \mathcal{D} , i.e., $\mathcal{C} \subseteq \mathcal{D}$. We further denote by \mathcal{C}_T be the constraint set in the T th iteration, then we have $\mathcal{C}_T \subseteq \mathcal{C}_{T+1}$. In T th iteration, we find a new constraint \mathbf{d}_{T+1} based on $\boldsymbol{\alpha}_T$, i.e., $-f(\boldsymbol{\alpha}_T, \mathbf{d}_{T+1}) = \max_{\mathbf{d} \in \mathcal{D}} -f(\boldsymbol{\alpha}_T, \mathbf{d})$. Define

$$\beta_T = \max_{1 \leq i \leq T} -f(\boldsymbol{\alpha}_T, \mathbf{d}_i) = \min_{\boldsymbol{\alpha} \in \mathcal{A}} \max_{1 \leq i \leq T} -f(\boldsymbol{\alpha}, \mathbf{d}_i) \quad (18)$$

and

$$\varphi_T = \min_{1 \leq j \leq T} -f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}), \quad (19)$$

where $-f(\boldsymbol{\alpha}_j, \mathbf{d}_{j+1}) = \max_{\mathbf{d} \in \mathcal{D}} -f(\boldsymbol{\alpha}_j, \mathbf{d})$. Similar in (Chen and Ye, 2008), we have the following theorem that indicates FGM gradually approaches to the optimal solution.

Theorem 2 *Let $(\boldsymbol{\alpha}^*, \theta^*)$ be the globally optimal solution pair of (16), $\{\beta_T\}$ and $\{\varphi_T\}$ as defined above, then:*

$$\beta_T \leq \theta^* \leq \varphi_T. \quad (20)$$

With the number of iteration k increasing, $\{\beta_T\}$ is monotonically increasing and the sequence $\{\varphi_T\}$ is monotonically decreasing.

Proof $\theta^* = \min_{\boldsymbol{\alpha} \in \mathcal{A}} \max_{\mathbf{d} \in \mathcal{D}} -f(\boldsymbol{\alpha}, \mathbf{d})$. For a fixed feasible $\boldsymbol{\alpha}$, we have $\max_{\mathbf{d} \in \mathcal{C}_T} -f(\boldsymbol{\alpha}, \mathbf{d}) \leq \max_{\mathbf{d} \in \mathcal{D}} -f(\boldsymbol{\alpha}, \mathbf{d})$, then $\min_{\boldsymbol{\alpha} \in \mathcal{A}} \max_{\mathbf{d} \in \mathcal{C}_T} -f(\boldsymbol{\alpha}, \mathbf{d}) \leq \min_{\boldsymbol{\alpha} \in \mathcal{A}} \max_{\mathbf{d} \in \mathcal{D}} -f(\boldsymbol{\alpha}, \mathbf{d})$, i.e. $\beta_T \leq \theta^*$. On the other hand, for

$\forall j = 1, \dots, k$, $-f(\alpha_j, \mathbf{d}_{j+1}) = \max_{\mathbf{d} \in \mathcal{D}} -f(\alpha_j, \mathbf{d})$, thus $(\alpha_j, -f(\alpha_j, \mathbf{d}_{j+1}))$ is a feasible solution pairs of (16). Then $\theta^* \leq -f(\alpha_j, \mathbf{d}_{j+1})$ for $j = 1, \dots, T$, and hence we have

$$\theta^* \leq \varphi_T = \min_{1 \leq j \leq T} -f(\alpha_j, \mathbf{d}_{j+1}).$$

With the number of iteration k increasing, the subset \mathcal{C}_T is monotonically increasing, so $\{\beta_T\}$ is monotonic increasing while $\{\varphi_T\}$ is monotonically decreasing. The proof is completed. \blacksquare

The following theorem further indicates that FGM can obtain the global solution to (16).

Theorem 3 *Assume that in each iteration of Algorithm 1, the QCQP subproblem (17) and the worst-case analysis in step 3 can be solved. After a finite number of steps, if there is no update of \mathcal{C}_T , i.e. $\mathcal{C}_{T+1} = \mathcal{C}_T$, FGM stops with the global optimal solution of (16); Otherwise, FGM generates a sequence of (α_T, \mathbf{d}_T) , which converges to global optimal solution (α^*, \mathbf{d}^*) of (16).*

Proof Firstly, we can measure the convergence of FGM by the gap difference of series $\{\beta_T\}$ and $\{\varphi_T\}$. Assume in T th iteration, there is no update of \mathcal{C}_T , i.e.

$$\mathbf{d}_{T+1} = \arg \max_{\mathbf{d} \in \mathcal{D}} -f(\alpha_T, \mathbf{d}) \in \mathcal{C}_T,$$

then $\mathcal{C}_T = \mathcal{C}_{T+1}$. So, we can prove that, in this case, (α_T, \mathbf{d}_T) is the globally optimal solution pair of (16). First, since $\mathcal{C}_T = \mathcal{C}_{T+1}$, in Algorithm 1, there will be no update of α , i.e. $\alpha_{T+1} = \alpha_T$. Then we have

$$-S(\alpha_T, \mathbf{d}_{T+1}) = \max_{\mathbf{d} \in \mathcal{D}} -f(\alpha_T, \mathbf{d}) = \max_{\mathbf{d} \in \mathcal{C}_T} -f(\alpha_T, \mathbf{d}) = \max_{1 \leq i \leq T} -S(\alpha_T, \mathbf{d}_i) = \beta_T, \quad (21)$$

and

$$\varphi_T = \min_{1 \leq j \leq T} -f(\alpha_j, \mathbf{d}_{j+1}) \leq \beta_T.$$

From Theorem 1, we know $\beta_T \leq \theta^* \leq \varphi_T$, then we obtain $\beta_T = \theta^* = \varphi_T$, and (α_T, \mathbf{d}_T) is the globally optimal solution pair of (16).

If the algorithm does not stop in finite steps, the algorithm will generate a sequence $(\alpha_T, \mathbf{d}_T) \in \mathcal{A} \times \mathcal{D}$, as \mathcal{A} and \mathcal{D} are compact set, so sequence (α_T, \mathbf{d}_T) converges to $(\alpha^*, \mathbf{d}^*) \in \mathcal{A} \times \mathcal{D}$. By above discussion, we know, if sequence \mathbf{d}_k converges, then the set \mathcal{C}_T will not have new update, and the algorithm finds global optimal solution pair (α^*, \mathbf{d}^*) . The proof is completed. \blacksquare

4. Efficient Worst-Case Analysis

The global convergence of FGM highly depends on whether or not the worst-case analysis in Algorithm 1 can be exactly and globally solved. In the following, the worst-case analysis of diverse cases will be discussed.

4.1 Feature Generation for Linear Feature Selection

In the linear feature selection, to conduct the worst-case analysis, we need to find the most violated \mathbf{d}_T in (16) at the T th iteration. When $|\mathcal{G}_j| = 1$, namely, there is no group definition regarding the features, we just need to solve the following maximization problem subjected to a linear constraint $\sum_{j=1}^m d_j \leq B$:

$$\max_{\mathbf{d} \in \mathcal{D}} \quad \frac{1}{2} \left\| \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}} \odot \boldsymbol{\lambda}) \right\|^2. \quad (22)$$

Notice that

$$\frac{1}{2} \left\| \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}} \odot \boldsymbol{\lambda}) \right\|^2 = \frac{1}{2} \left\| \sum_{i=1}^n (\alpha_i y_i \mathbf{x}_i) \odot \sqrt{\mathbf{d}} \odot \boldsymbol{\lambda} \right\|^2 = \frac{1}{2} \sum_{j=1}^m \omega_j^2 \lambda_j^2 d_j,$$

where $\boldsymbol{\omega} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$. Let $c_j = \omega_j^2 \lambda_j^2$, then the globally optimal solution of this problem can be easily obtained without any numeric optimization solver. That is, it can be solved by first sorting c_j 's, and then setting the corresponding to d_j to 1 of the first B number of c_j 's and the rests to 0. And the selected features will form a feature group of size B .

4.2 Feature Generation for Group Feature Selection

When dealing with the group selection, firstly suppose we have the non-overlapping group structure $\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_p\}$ such that $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \forall i, j$, and a group scaling vector $\mathbf{d} = [0, \dots, 0] \in \mathbb{R}^p$. Different from the linear feature problem, we need to solve:

$$\max_{\mathbf{d} \in \mathcal{D}} \quad \sum_{j=1}^p d_j \lambda_j^2 \left\| \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{i\mathcal{G}_j} \right\|^2 = \max_{\mathbf{d} \in \mathcal{D}} \sum_{j=1}^p d_j \lambda_j^2 \boldsymbol{\omega}'_{\mathcal{G}_j} \boldsymbol{\omega}_{\mathcal{G}_j}, \quad (23)$$

where $\boldsymbol{\omega}_{\mathcal{G}_j} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{i\mathcal{G}_j}$ is for group \mathcal{G}_j . Let $c_{\mathcal{G}_j} = \lambda_j^2 \boldsymbol{\omega}'_{\mathcal{G}_j} \boldsymbol{\omega}_{\mathcal{G}_j}$, then the globally optimal solution of this problem can be easily obtained by first sorting $c_{\mathcal{G}_j}$'s, and then setting d_j corresponding to the first B number of $c_{\mathcal{G}_j}$'s to 1 and the rests to 0. Notice that if $|\mathcal{G}_j| = 1$ for $j = 1, \dots, p$, we have $p = m$, $\mathcal{G} = \{1, \dots, m\}$ and $c_{\mathcal{G}_j} = \omega_j^2$. Hence, we can unify the worst-case analysis of linear feature selection and group selection as in Algorithm 2 by using the notations of groups, which returns an active constraint $\mathbf{d} \in \mathcal{D}$. Different from the linear case, in the group case, \mathbf{d} will include B groups of features, which will form a **super group**.

Algorithm 2 Feature Inference by Worst Case Analysis.

Given the dual variable $\boldsymbol{\alpha}$, B , the additional scaling factor $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_p]$, a zero column vector $\mathbf{c} \in \mathbb{R}^p$, training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and group set $\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_p\}$.

- 1: Calculate $\boldsymbol{\omega} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$.
 - 2: Calculate the score \mathbf{c} , where $c_{\mathcal{G}_j} = \lambda_j^2 \boldsymbol{\omega}'_{\mathcal{G}_j} \boldsymbol{\omega}_{\mathcal{G}_j}$.
 - 3: Sort and find the B largest $c_{\mathcal{G}_j}$'s.
 - 4: Set d_j corresponding to the first B number of $c_{\mathcal{G}_j}$'s to 1 and the rests to 0.
 - 5: Return \mathbf{d} .
-

The above worst-case analysis algorithm can be also applied to the overlapping groups without any modifications. The only difference is that, for non-overlapping groups, we have $p < m$, while for the overlapping groups, p can be larger than m . And the finding of the B most active groups requires $O(mn + p \log(B))$, for which the first term indicates the computational cost to calculate ω and the latter term indicates the complexity of the calculation and sorting of $c_{\mathcal{G}_j}$.

4.3 Feature Generation with Groups of Special Structures

As the complexity for worst-case analysis is $O(mn + p \log(B))$, the second term can be negligible if p is small. However, when the groups are organized in graph or tree structures, p becomes very large and the computational cost of Algorithm 2 will be very expensive (Jenatton et al., 2011). Notice that in Algorithm 2, we can maintain a cache of the score and indices of the B features with the largest score. Let c_B^{\min} be the smallest score in the cache, after the calculation of $c_{\mathcal{G}_j}$, we can dynamically do the sorting and then update the cache by comparing $c_{\mathcal{G}_j}$ and c_B^{\min} . Then if groups follows the tree-structures defined in Definition 1 (Jenatton et al., 2011), the computational cost can reduce to $O(nm + \log(p) \log(B))$, as will be shown in Proposition 2.

Proposition 2 *Given a set of groups $\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_p\}$ organized as a tree structure in Definition 1, let c_B^{\min} be the smallest score of the B largest scores for the current search, suppose $\mathcal{G}_h \subseteq \mathcal{G}_g$, then \mathcal{G}_h will not be selected if $\lambda_h^2 \|\omega_{\mathcal{G}_g}\|^2 < c_B^{\min}$. Particularly, \mathcal{G}_g and all its decedent $\mathcal{G}_h \subseteq \mathcal{G}_g$ will not be selected if $\lambda_{\max}^2 \|\omega_{\mathcal{G}_g}\|^2 < c_B^{\min}$, where $\lambda_{\max} = \max\{\lambda_h, \mathcal{G}_h \subseteq \mathcal{G}_g\}$.*

Proof Notice that $c_g = \lambda_g^2 \omega'_{\mathcal{G}_g} \omega_{\mathcal{G}_g}$. For its decedent \mathcal{G}_h , $c_h = \lambda_h^2 \omega'_{\mathcal{G}_h} \omega_{\mathcal{G}_h} \leq \lambda_h^2 \omega'_{\mathcal{G}_g} \omega_{\mathcal{G}_g} \leq \lambda_{\max}^2 \omega'_{\mathcal{G}_g} \omega_{\mathcal{G}_g}$. Then \mathcal{G}_h will not be selected if $\lambda_h^2 \|\omega_{\mathcal{G}_g}\|^2 < c_B^{\min}$. In addition, if $\lambda_{\max}^2 \omega'_{\mathcal{G}_g} \omega_{\mathcal{G}_g} < c_B^{\min}$, all the descendants of \mathcal{G}_g will not be selected. Particularly, if λ_h for all groups equals to 1, then all the decedent of \mathcal{G}_g will not be selected if $\omega'_{\mathcal{G}_g} \omega_{\mathcal{G}_g} < c_B^{\min}$. ■

With the above property, the computational cost of the $O(nm)$ term can be much reduced if we calculate $\omega_{\mathcal{G}_h}$ dynamically in the sense that there is no need to calculate $\omega_{\mathcal{G}_h}$ of those non-informative groups as well as their descendants.

Remark 2 *Suppose that the cutting plane algorithm stops after T iterations, then $B \leq \text{Card}(\mathbf{d}^*) \leq TB$. Notice that, from the worst-case analysis, $\sqrt{\mathbf{d}_t} \in \{\mathbf{d} \mid \sum_{j=1}^m d_j = B, d_j \in \{0, 1\}, j = 1, \dots, m\}$. With $\sum_{t=1}^T \mu_t^* = 1$, clearly we have $\mathbf{d}^* \in \mathcal{D} = \{\mathbf{d} \mid \sum_{j=1}^m d_j \leq B, d_j \in [0, 1], j = 1, \dots, m\}$. Because $d_i^* \in [0, 1]$, we have $\text{Card}(\mathbf{d}^*) \geq B$, which means that we may find more than B features. On the other hand, suppose the cutting plane algorithm stops after T iterations, then $\text{Card}(\mathbf{d}^*) \leq TB$. Finally we have $B \leq \text{Card}(\mathbf{d}^*) \leq TB$. It indicates that we can always constrain the number of features or groups in the range of $[B, TB]$.*

4.4 Nonlinear Feature Selection with Kernels

FGM can be extended to do nonlinear feature selection by using the kernel tricks. Specifically, let $\phi(\mathbf{x} \odot \sqrt{\mathbf{d}})$ denote the nonlinear feature mapping that maps the features into a high dimensional feature space, where $\mathbf{0} \preceq \mathbf{d} \preceq \mathbf{1}$ is the feature scaling vector imposed on

features. By substituting the linear term $(\mathbf{x} \odot \sqrt{\mathbf{d}})$ in (6) with the nonlinear feature map $\phi(\mathbf{x} \odot \sqrt{\mathbf{d}})$, we obtain the kernel version of FGM with the similar deduction as in linear case. Alternatively, we need to solve the following infinite kernel learning problem (Gehler and Nowozin, 2008):

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\boldsymbol{\mu} \in \mathcal{M}} \sum_{\mathbf{d}_t \in \mathcal{D}} \mu_t f(\boldsymbol{\alpha}, \mathbf{d}_t) = \min_{\boldsymbol{\mu} \in \mathcal{M}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} -\frac{1}{2}(\boldsymbol{\alpha} \odot \mathbf{y})' \left(\sum_{\mathbf{d}_t \in \mathcal{D}} \mu_t \mathbf{K}_t + \frac{1}{C} I \right) (\boldsymbol{\alpha} \odot \mathbf{y}), \quad (24)$$

where \mathbf{K}_t is defined by $\phi(\mathbf{x} \odot \mathbf{d}_t)' \phi(\mathbf{x} \odot \mathbf{d}_t)$ with $\mathbf{d}_t \in \mathcal{D}$. This problem can be also solved by the Algorithm 1. One critical issue is to conduct the worst-case analysis, where we need to solve the following optimization problem:

$$\max_{\mathbf{d} \in \mathcal{D}} \frac{1}{2} \left\| \sum_{i=1}^n \alpha_i y_i \phi(\mathbf{x}_i \odot \mathbf{d}) \right\|^2 = \max_{\mathbf{d} \in \mathcal{D}} \frac{1}{2} (\boldsymbol{\alpha} \odot \mathbf{y})' \tilde{\mathbf{K}}_d (\boldsymbol{\alpha} \odot \mathbf{y}), \quad (25)$$

where $\tilde{\mathbf{K}}_d$ is short for $\mathbf{K}(\mathbf{x}_i \odot \mathbf{d}, \mathbf{x}_j \odot \mathbf{d})$, which is defined by $\phi(\mathbf{x} \odot \mathbf{d})' \phi(\mathbf{x} \odot \mathbf{d})$. For general kernels such as Gaussian kernels, this problem is NP-hard to solve. But if \mathbf{K} can be decomposed as p additive kernels, namely, $\mathbf{K} = \sum_{k=1}^p \mathbf{K}_k$, where \mathbf{K}_k is a sub-kernel that constructed by one feature or part of the features (Maji and Berg, 2009), the worst-case analysis can be efficiently and globally solved. Typical additive kernels includes the intersection kernel (Maji and Berg, 2009) and ANOVA kernel $k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^m \exp(\theta(x_i - y_i)^2)^\varpi$, where θ and ϖ are the parameters (Bach, 2009). Taking the histogram intersection kernel as an example, we can easily verify that the worst-case analysis can be easily and globally solved (Maji and Berg, 2009).

Remark 3 *FGM can globally converge regarding the intersection kernel (Maji and Berg, 2009).*

Proof The global convergence of FGM depends on whether or not the worst-case analysis can be globally solved. We will prove that the worst-case analysis for general intersection kernel can be easily and globally solved. Notice that the general intersection kernel is defined as:

$$k(\mathbf{x}, \mathbf{z}, \beta) = \sum_{k=1}^m \min\{|x_k|^\beta, |z_k|^\beta\}, \quad (26)$$

where $\beta > 0$ is a kernel parameter. When $\beta = 1$, it is reduce to the well-known Histogram Intersection Kernel (HIK). By introducing the feature scaling vector \mathbf{d} to the features,

$k(\mathbf{x}, \mathbf{z}, \beta, \mathbf{d}) = \sum_{k=1}^m d_k \min\{|x_k|^\beta, |z_k|^\beta\}$. Hence, the subproblem for HIK kernel is:

$$\begin{aligned}
& \max_{\mathbf{d} \in \mathcal{D}} \left\| \sum_{i=1}^n \alpha_i y_i \phi(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \right\|^2 \\
&= \max_{\mathbf{d} \in \mathcal{D}} \sum_{k=1}^m d_k \left(\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \min\{|x_k^i|^\beta, |x_k^j|^\beta\} \right) \\
&= \max_{\mathbf{d} \in \mathcal{D}} \sum_{k=1}^m d_k c_k,
\end{aligned} \tag{27}$$

where $c_k = (\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \min\{|x_k^i|^\beta, |x_k^j|^\beta\})$. Obviously, we can easily and exactly solve (27) via sorting c_k . Hence FGM with intersection kernel can retain the global convergence. \blacksquare

Recently, Bach (2009) proposed novel additive kernel construction scheme, where each sub-kernel is defined by each vertex of the tree or graph structures on features. However, how to construct the tree-structure or graph-structure for general problems is still an open issue. In our experimental verifications, we simply use the explicit feature mapping to generate the groups (Vedaldi and Zisserman, 2010). The random feature and HIK feature expansion are typical examples, which can help to solve large-scale problems (Vedaldi and Zisserman, 2010; Wu et al., 2010). For additive kernels where each kernel is constructed by part of features, we can expand each sub-kernel by use its nonlinear feature mappingd. Then the feature selection becomes a group selection problem, which will be further detailed in Section 7.

5. Subproblem Learning

5.1 Learning in the Dual

The convergence of the cutting plane algorithm highly depends on solving problem (17) exactly. At the T th iteration in Algorithm 1, a new \mathbf{d}_t will be included into \mathcal{C} , and we have $|\mathcal{C}| = T$. For convenience, we make some notations as follows. As each \mathbf{d}_t select B features or feature groups, let $\boldsymbol{\lambda}_t \in \mathbb{R}^B$ be the data scaling vector of the selected features or groups. For linear case, we can define $\mathbf{X}_t \triangleq [(\mathbf{x}_{1t} \odot \boldsymbol{\lambda}_t)', \dots, (\mathbf{x}_{nt} \odot \boldsymbol{\lambda}_t)']'$ as the data matrix of the selected features, where $\mathbf{x}_{it}, i = 1, \dots, n$ denotes the i th instance with the features selected by \mathbf{d}_t . For group case, as we have $f(\mathbf{w}) = \mathbf{w}'\mathbf{x} = \sum_{j=1}^p \lambda_j \sqrt{d_j} \mathbf{w}'_{\mathcal{G}_j} \mathbf{x}_{\mathcal{G}_j}$, we can augment $\lambda_j \in \mathbb{R}$ for group \mathcal{G}_j as $\hat{\boldsymbol{\lambda}}_j = [\lambda_j, \dots, \lambda_j]' \in \mathbb{R}^{|\mathcal{G}_j|}$ to weight each feature and define $\hat{\mathbf{X}}^j \triangleq [(\mathbf{x}_{1j} \odot \hat{\boldsymbol{\lambda}}_j)', \dots, (\mathbf{x}_{nj} \odot \hat{\boldsymbol{\lambda}}_j)']'$ as the data matrix of the selected features indexed by \mathcal{G}_j . As a result, we can further define \mathbf{X}_t by concatenating all $\hat{\mathbf{X}}^j$ as the data matrix of the selected features by \mathbf{d}_t on groups.

Let $\mu_t \geq 0$ be the dual variable for each constraint defined by \mathbf{d}_t in (17), its Lagrangian can be written as:

$$\mathcal{L}(\theta, \boldsymbol{\alpha}, \boldsymbol{\mu}) = -\theta - \sum_{t=1, \mathbf{d}_t \in \mathcal{C}}^T \mu_t (f(\boldsymbol{\alpha}, \mathbf{d}_t) - \theta). \quad (28)$$

Setting its derivative w.r.t. θ to zero, we have $\sum \mu_t = 1$. Furthermore, let $\boldsymbol{\mu}$ be the vector of μ_t 's, and $\mathcal{M} = \{\boldsymbol{\mu} | \sum \mu_t = 1, \mu_t \geq 0\}$ be its domain, and then the Lagrangian $\mathcal{L}(\theta, \boldsymbol{\alpha}, \boldsymbol{\mu})$ can be rewritten as:

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\boldsymbol{\mu} \in \mathcal{M}} - \sum_{\mathbf{d}_t \in \mathcal{C}} \mu_t f(\boldsymbol{\alpha}, \mathbf{d}_t) = \min_{\boldsymbol{\mu} \in \mathcal{M}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} - \frac{1}{2} (\boldsymbol{\alpha} \odot \mathbf{y})' \left(\sum_{\mathbf{d}_t \in \mathcal{C}} \mu_t \mathbf{X}_t \mathbf{X}_t' + \frac{1}{C} \mathbf{I} \right) (\boldsymbol{\alpha} \odot \mathbf{y}), \quad (29)$$

where \mathbf{X}_t is defined above and the equality holds due to the fact that the objective function is concave in $\boldsymbol{\alpha}$ and convex in $\boldsymbol{\mu}$. Obviously, equation (29) can be regarded as a MKL problem (Lanckriet et al., 2004; Rakotomamonjy et al., 2008), where the kernel matrix $\sum_{\mathbf{d}_t \in \mathcal{C}} \mu_t \mathbf{X}_t \mathbf{X}_t'$ to be learned is a convex combination of $|\mathcal{C}| = T$ base kernel matrices $\mathbf{X}_t \mathbf{X}_t'$, each of which is constructed from a feature (group) selection vector $\mathbf{d}_t \in \mathcal{C}$. Several efficient MKL approaches have been proposed to solve such problems in recent years: Lanckriet et al. (2004); Rakotomamonjy et al. (2007, 2008). Following SimpleMKL algorithm, we can solve (29) iteratively (Rakotomamonjy et al., 2008). First, we fix the coefficients $\boldsymbol{\mu}$ of the base kernel matrices and solve the SVM's dual:

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} - \frac{1}{2} (\boldsymbol{\alpha} \odot \mathbf{y})' \left(\sum_{t=1}^T \mu_t \mathbf{X}_t \mathbf{X}_t' + \frac{1}{C} \mathbf{I} \right) (\boldsymbol{\alpha} \odot \mathbf{y}). \quad (30)$$

Then, we fix $\boldsymbol{\alpha}$ and use the reduced gradient method to update $\boldsymbol{\mu}$. These two steps alternates until convergence. When solving (30), we can cater for the state-of-art LIBlinear solver (Hsieh et al., 2008), in which a primal-dual coordinate descent method is used to solve the quadratic programming problem.

5.2 Learning in the Primal

Although we can solve the subproblem (17) by SimpleMKL, it is not scalable for large-scale problems. Specifically, in SimpleMKL, many times of the SVM training are needed to calculate the sub-gradient, making SimpleMKL inefficient. Furthermore, due to the numerical issues, it is very expensive to obtain an accurate SVM model, thus the precision of the sub-gradient method will be limited (Nedic and Ozdaglar, 2009). We can also apply the SQP methods to solve the minimax SimpleMKL problem (Pee and Royset, 2010). However, as pointed in (Pee and Royset, 2010), it may need more run-time complexity.

Regarding the above issues, in the following, we propose to solve (29) in the primal. In equation (29), with the definition of \mathbf{X}_t above, without loss of generality, denote by $\boldsymbol{\omega}_t$ the weights of the associated features of \mathbf{X}_t , and let $\boldsymbol{\omega}$ be a supervector concatenating all $\boldsymbol{\omega}_t$'s, namely, $\boldsymbol{\omega} = [\boldsymbol{\omega}_1', \dots, \boldsymbol{\omega}_T']'$. Then we can show that the primal form of the MKL subproblem (29) can be expressed as following ℓ_1^2 -regularized problem.

Theorem 4 *The primal form of the MKL subproblem (29) can be expressed as following ℓ_1^2 -regularized problem:*

$$\min_{\boldsymbol{\omega}} \quad \frac{1}{2} \left(\sum_{t=1}^T \|\boldsymbol{\omega}_t\| \right)^2 + p(\boldsymbol{\omega}), \quad (31)$$

where $p(\boldsymbol{\omega}) = \frac{C}{2} \sum_{i=1}^n \xi_i^2$ is for the square hinge loss with $\xi_i = \max(1 - y_i \sum_t \boldsymbol{\omega}'_t \mathbf{x}_{it}, 0)$, $p(\boldsymbol{\omega}) = C \sum_{i=1}^n \log(1 + \exp(\xi_i))$ is for the logistic loss with $\xi_i = -y_i \sum_{t=1}^T \boldsymbol{\omega}'_t \mathbf{x}_{it}$, and \mathbf{x}_{it} denotes the i th instance of \mathbf{X}_t . Furthermore, once we obtain the optimal solution, we can recover the dual variables for the square-hinge loss by $\alpha_i = C\xi_i^*$ while recover the dual variables for the logistic loss by $\alpha_i = \frac{C \exp(\xi_i)}{1 + \exp(\xi_i^*)}$.

Proof By applying the conic duality theory, the proof parallels the results in (Bach et al., 2004). Let $\Omega(\boldsymbol{\omega}) = \frac{1}{2} (\|\boldsymbol{\omega}_t\|)^2$ and define the cone $\mathcal{Q}_B = \{(\mathbf{u}, v) \in \mathbb{R}^{B+1}, \|\mathbf{u}\|_2 \leq v\}$. Furthermore, let $z_t = \|\boldsymbol{\omega}_t\|$, then we have $\Omega(\boldsymbol{\omega}) = \frac{1}{2} \left(\sum_{t=1}^T \|\boldsymbol{\omega}_t\| \right)^2 = \frac{1}{2} z^2$ with $z = \sum_{t=1}^T z_t$. Apparently, we have $z_t \geq 0$ and $z \geq 0$. Finally, problem (31) can be transformed into the following form:

$$\begin{aligned} \min_{z, \boldsymbol{\omega}} \quad & \frac{1}{2} z^2 + p(\boldsymbol{\omega}), \\ \text{s.t.} \quad & \sum_{t=1}^T z_t \leq z, \quad (\boldsymbol{\omega}_t, z_t) \in \mathcal{Q}_B. \end{aligned} \quad (32)$$

By introducing the Lagrangian dual variable $\boldsymbol{\alpha}$, γ and $\boldsymbol{\zeta}_t$, the Lagrangian function of (31) regarding the squared hinge loss can be written as:

$$\begin{aligned} \mathcal{L}(z, \boldsymbol{\omega}, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \mu) &= \frac{1}{2} z^2 + \frac{C}{2} \sum_{i=1}^n \xi_i^2 - \sum_{i=1}^n \alpha_i (\xi_i - y_i \sum_t \boldsymbol{\omega}'_t \mathbf{x}_{it} + 1) \\ &+ \gamma \left(\sum_{t=1}^T z_t - z \right) - \sum_{t=1}^T (\boldsymbol{\zeta}'_t \boldsymbol{\omega}_t + \mu_t z_t), \end{aligned} \quad (33)$$

where $\boldsymbol{\omega} = [\boldsymbol{\omega}'_1, \dots, \boldsymbol{\omega}'_T]'$. Hence the KKT condition can be obtained as

$$\begin{aligned} \nabla_z \mathcal{L} = z - \gamma &= 0 & \Rightarrow z &= \gamma; \\ \nabla_{z_t} \mathcal{L} = \gamma - \mu_t &= 0 & \Rightarrow \mu_t &= \gamma; \\ \nabla_{\boldsymbol{\omega}_t} \mathcal{L} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{it} - \boldsymbol{\zeta}_t &= 0 & \Rightarrow \boldsymbol{\zeta}_t &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{it}; \\ \nabla_{\xi_i} \mathcal{L} = C\xi_i - \alpha_i &= 0 & \Rightarrow \xi_i &= \frac{\alpha_i}{C} \quad \text{or} \quad \alpha_i = C\xi_i; \\ \|\boldsymbol{\zeta}_t\| &\leq \mu_t & \Rightarrow \|\boldsymbol{\zeta}_t\| &\leq \gamma. \end{aligned}$$

By substituting all the above results into (33), we obtain

$$\mathcal{L}(z, \omega, \alpha, \gamma, \zeta, \mu) = -\frac{1}{2}\gamma^2 - \frac{1}{2C}\alpha'\alpha. \quad (34)$$

Hence the dual problem of the ℓ_1^2 -regularized problem regarding the squared hinge loss can be written as:

$$\begin{aligned} \max_{\gamma, \alpha} \quad & -\frac{1}{2}\gamma^2 - \frac{1}{2C}\alpha'\alpha \\ \text{s.t.} \quad & \left\| \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{it} \right\| \leq \gamma, \quad t = 1, \dots, T, \\ & \alpha_i \geq 0, \quad i = 1, \dots, n. \end{aligned} \quad (35)$$

Let $\theta = \frac{1}{2}\gamma^2 + \frac{1}{2C}\alpha'\alpha$ and $f(\alpha, \mathbf{d}_t) = \frac{1}{2}\|\zeta_t\|^2 + \frac{1}{2C}\alpha'\alpha$ which is defined by \mathbf{d}_t , then we have

$$\begin{aligned} \max_{\theta, \alpha} \quad & -\theta \\ \text{s.t.} \quad & f(\alpha, \mathbf{d}_t) \leq \theta, \quad t = 1, \dots, T, \\ & \alpha_i \geq 0, \quad i = 1, \dots, n. \end{aligned} \quad (36)$$

which indeed is in the form of problem (17) by letting \mathcal{A} be the domain of α . This completes the proof and brings the connection between the primal and dual formulation.

For logistic regression problem, by defining $0 \log(0) = 0$, with the similar derivation above, we can obtain the dual problem regarding the logistic loss as follows:

$$\begin{aligned} \max_{\gamma, \alpha} \quad & -\frac{1}{2}\gamma^2 - \sum_{i=1}^n (C - \alpha_i) \log(C - \alpha_i) - \sum_{i=1}^n \alpha_i \log(\alpha_i) \\ \text{s.t.} \quad & \left\| \sum_{i=1}^n \alpha_i y_i \mathbf{x}_{it} \right\| \leq \gamma, \quad t = 1, \dots, T, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, n. \end{aligned} \quad (37)$$

Again, we can easily recover the dual variable α by $\alpha_i = \frac{C \exp(\xi_i)}{1 + \exp(\xi_i)}$. This completes the proof. \blacksquare

Remark 4 From Theorem 4, we can efficiently solve the MKL subproblem (29) in its primal. Notice that in Algorithm 1, we need to calculate the dual variable α to do the worst-case analysis. Fortunately, for the squared hinge loss and logistic loss, we can easily recover the dual variable α via the training errors ξ . However, this property does not hold for the hinge loss because the dual variables cannot be recovered from the primal variables.

Once establishing the connection between the dual and primal form of the QCQP subproblem (29), we can solve its primal form (31). Specifically, we can apply the accelerated

proximal gradient (APG) method to efficiently solve it (Toh and Yun, 2009). For the convenience of presentation, let $\Omega(\omega) = \frac{1}{2} \left(\sum_{t=1}^T \|\omega_t\| \right)^2$ and define

$$F(\omega) = \Omega(\omega) + p(\omega), \quad (38)$$

Apparently, $F(\omega)$ is non-smooth and non-separable. The key procedure of APG is to construct the following quadratic approximation of $F(\omega)$ at the point \mathbf{v} by

$$\begin{aligned} Q_\tau(\omega, \mathbf{v}) &= p(\mathbf{v}) + \langle \nabla p(\mathbf{v}), \omega - \mathbf{v} \rangle + \Omega(\omega) + \frac{\tau}{2} \|\omega - \mathbf{v}\|^2 \\ &= \frac{\tau}{2} \|\omega - \mathbf{g}\|^2 + \Omega(\omega) + p(\mathbf{v}) - \frac{1}{2\tau} \|\nabla p(\mathbf{v})\|^2, \end{aligned} \quad (39)$$

where $\tau > 0$ and $\mathbf{g} = \mathbf{v} - \frac{1}{\tau} \nabla p(\mathbf{v})$. Then, we need to solve the following Moreau Projection problem (Martins et al., 2010):

$$\min_{\omega} \frac{\tau}{2} \|\omega - \mathbf{g}\|^2 + \Omega(\omega). \quad (40)$$

Although $\Omega(\omega)$ in our problem is non-separable, according to (Martins et al., 2010), problem (40) has a unique closed-form solution given by Algorithm 3. Specifically, for the convenience of presentation, write \mathbf{g} as $\mathbf{g} = [\mathbf{g}'_1, \dots, \mathbf{g}'_T]$ and let $\mathbf{u} = [\|\mathbf{g}_1\|, \dots, \|\mathbf{g}_T\|]' \in \mathbb{R}^T$, where \mathbf{g}_t corresponds to ω_t for $t = 1, \dots, T$, then the following proposition holds true.

Proposition 3 *Let $S_\tau(\mathbf{g})$ be the optimal solution to problem (40) and $\mathbf{o} \in \mathbb{R}^T$ be an intermediate variable, then $S_\tau(\mathbf{g})$ is unique and can be calculated as follows,*

$$S_\tau(\mathbf{g}_t) = \begin{cases} \frac{o_t}{\|\mathbf{g}_t\|} \mathbf{g}_t, & \text{if } o_t > 0, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \quad (41)$$

The intermediate vector \mathbf{o} can be calculated via a soft-threshold operator $\text{soft}(\mathbf{u}, \varsigma)$ defined by:

$$o_t = [\text{soft}(\mathbf{u}, \varsigma)]_t = \begin{cases} u_t - \varsigma, & \text{if } u_t > \varsigma, \\ 0, & \text{Otherwise,} \end{cases}$$

where the threshold value ς can be calculated in Step 3 of Algorithm 3.

Algorithm 3 Moreau Projection $S_\tau(\mathbf{v})$.

Given input \mathbf{g} , $s = \frac{1}{\tau}$ and the number of kernels T .

1: Calculate $\hat{u}_t = \|\mathbf{g}_t\|$ for all $t = 1, \dots, T$.

2: Sort $\hat{\mathbf{u}}$ such that $\mathbf{u}_{(1)} \geq \dots \geq \mathbf{u}_{(T)}$.

3: Find $\rho = \max \left\{ j | u_j - \frac{s}{1+js} \sum_{i=1}^j u_i > 0, j = 1, \dots, T \right\}$.

4: Calculate the threshold value $\varsigma = \frac{s}{1+\rho s} \sum_{i=1}^{\rho} u_i$.

5: Compute $\mathbf{o} = \text{soft}(\hat{\mathbf{u}}, \varsigma)$.

6: Compute and output $S_\tau(\mathbf{g})$.

Proof The proof can be adapted from (Martins et al., 2010). ■

Algorithm 4 Accelerated Proximal Gradient Method for Solving the ℓ_1^2 problem.

Initialization: Initialize the Lipschitz constant L_T and set $\omega^{-1} = \omega^0$ by warm start, $\tau_0 = L_T$, $\eta \in (0, 1)$, parameter $\varrho^{-1} = \varrho^0 = 1$ and $k = 0$.

1: Set $\mathbf{v}^k = \omega^k + \frac{\varrho^{k-1}-1}{\varrho^k}(\omega^k - \omega^{k-1})$.

2: Set $\tau = \eta\tau_k$.

For $j = 0, 1, \dots$,

Set $\mathbf{g} = \mathbf{v}^k - \frac{1}{\tau}\nabla p(\mathbf{v}^k)$, compute $S_\tau(\mathbf{g})$.

if $F(S_\tau(\mathbf{g})) \leq Q(S_\tau(\mathbf{g}), \mathbf{v}^k)$,

set $\tau_k = \tau$, stop;

else

$\tau = \min\{\eta^{-1}\tau, L_T\}$.

End

end

3: Set $\omega^{k+1} = S_{\tau_k}(\mathbf{g})$.

4: Compute $\varrho^{k+1} = \frac{1+\sqrt{(1+4(\varrho^k)^2)}}{2}$. Let $k = k + 1$.

5: Quit if stopping condition achieves. Otherwise, go to step 1.

Remark 5 For the Moreau Projection in Algorithm 3, a sorting with $O(T)$ is needed. In our problem settings, T is usually very small. Hence the Moreau Projection can be calculated very efficiently, which will be particularly important when p or m is hugely large. In addition, $S_\tau(\mathbf{g}_t)$ either equals to $\mathbf{0}$ or parallels \mathbf{g}_t , which will be very useful for accelerating the convergence speed by using a caching technique.

With the calculation of $S_\tau(\mathbf{g})$ in Algorithm 3, the details of the APG algorithm for solving the primal form of QCQP subproblem at the T th iteration is presented in Algorithm 4, where ω^{-1} and ω^0 be the internal solutions in the standard APG algorithm (Toh and Yun, 2009). In addition, L_T is the guess of the Lipschitz constant of the loss function regarding all the selected features at iteration T in Algorithm 1 and it is hard to compute for large-scale problems. In our settings, it is estimated by $L_0 = 0.1nC$ for the first iteration of Algorithm 1. For $T > 0$, L_T can be estimated by warm start $L_T = \eta^2\tau_k$, where τ_k is obtained from Algorithm 4 at the $(T-1)$ th outer cutting plane iteration. The internal solutions ω^{-1} and ω^0 can be also initialized by warm-start, which will be discussed later. As a subroutine of Algorithm 1, Algorithm 4 will be called at each outer cutting plane iteration T once an active constraint is added. Regarding the APG algorithm, the following convergence rate is guaranteed.

Theorem 5 (Ji et al., 2009) Let $\{\omega^{k+1}\}$, $\{\mathbf{v}^k\}$ and $\{\varrho^k\}$ be the sequences generated by APG and L_T be the Lipschitz constant of $p(\omega)$ at iteration T in Algorithm 1, for any $k \geq 1$, we have

$$F(\omega^k) - F(\omega^*) \leq \frac{2L_T\|\omega^0 - \omega^*\|^2}{\eta(k+1)^2}.$$

Proof The proof can be adapted from (Ji et al., 2009; Toh and Yun, 2009). ■

5.3 Implementation Issues

Thanks to the feature generating scheme, several implementation techniques, including the cache technique and warm start, can be taken to improve the training speed of FGM.

5.3.1 CACHE TECHNIQUE

Cache technique is one of the most important strategies widely used in many large-scale machine learning algorithms, such as the kernel SVM solver in LIBSVM (Fan et al., 2005). Different from LIBSVM, the cache technique used in FGM cache the selected features rather than the kernel entries. Here only the implementation issues for linear feature selection are discussed. Similar discussions can be applied to the group selections.

In gradient-based learning algorithms (for both ℓ_1 and ℓ_2 regularized problems), one needs to calculate the term $\mathbf{w}'\mathbf{x}_i$ for each instance to compute the gradient of the loss functions, which will cost $O(mn)$ for traditional methods. Unlike existing gradient-based learning methods that compute the gradient or sub-gradient based on all the input features, now in FGM the gradient computations in APG is based on the informative features only. In other words, once the indices of the informative features are generated via the worst-case analysis, we can cache these features to improve the training efficiency by accelerating the feature entry retrievals, which will need additional $O(TBn)$ memory storage. Hence, we can significantly reduce the original operation complexity from $O(nm)$ to $O(TBn)$, where usually $TB \ll m$ for high dimensional problems. This technique is particularly important in the feature selection with the explicit feature mappings, where the expanded dataset cannot be loaded into the main memory. In this case, instead of storing the whole expanded dataset, we need only to store the TB informative features and do the optimization efficiently without the needs of scanning of the whole dataset.

The cache technique can be also used to accelerate the line search of APG implementation. Notice that in APG, a line search is needed to find a suitable step size to make a sufficient decrease of the objective value. When performing the line search, we may need to calculate $p(S_\tau(\mathbf{g}))$ many times, which can be very expensive. Let $\boldsymbol{\omega} = S_\tau(\mathbf{g})$ with T groups $\boldsymbol{\omega} = [\boldsymbol{\omega}'_1, \dots, \boldsymbol{\omega}'_T]'$. As from Algorithm 3, $\mathbf{g} = \mathbf{v} - \frac{1}{\tau}\nabla p(\mathbf{v})$ and $S_\tau(\mathbf{g}_t) = c_t\mathbf{g}_t = c_t(\mathbf{v}_t - \frac{1}{\tau}\nabla p(\mathbf{v}_t))$ with $c_t \geq 0$ for $t = 1, \dots, T$. Hence the basic calculation $\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i$ of $p(\boldsymbol{\omega})$ follows

$$\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i = \sum_{i=1}^n \left(\sum_{t=1}^T \boldsymbol{\omega}'_t \mathbf{x}_{it} \right) = \sum_{i=1}^n \left(\sum_{t=1}^T c_t \left(\boxed{\mathbf{v}'_t \mathbf{x}_{it}} - \frac{1}{\tau} \boxed{\nabla p(\mathbf{v}_t)' \mathbf{x}_{it}} \right) \right). \quad (42)$$

According to the above calculation rule, we can make a fast computation of $\sum_{i=1}^n \boldsymbol{\omega}'\mathbf{x}_i$ by caching $\mathbf{v}'_t \mathbf{x}_{it}$ and $\nabla p(\mathbf{v}_t)' \mathbf{x}_{it}$ for each feature group t for each instance \mathbf{x}_i . With this cache scheme, no matter how many line search steps will be cost, it only needs to scan the selected features once. Hence the computation cost can be much reduced.

The cache strategy can also be used in the worst-case analysis in the explicit mapping feature selections, where we still need to compute the feature score $\mathbf{c} = \sum_{i=1}^n \alpha_i \mathbf{y}_i \mathbf{x}_i$, where

\mathbf{x}_i is the instance in the feature space. As will be discussed in Section 6.4, the dimension of the expanded \mathbf{x} can exceed 10^{12} , we cannot store \mathbf{c} explicitly. To address this problem, we can partition the dataset along the dimension into small data groups and calculate the score of each data group independently. As we only need to store the scores of the B features with the largest scores, without storing the full explicit feature mapping, the worst-case analysis can be efficiently done in a sequential way by caching the indices of the B best features as well as the associated scores.

5.3.2 WARM START

Warm start strategy is another technique that can greatly improve the efficiency of FGM. From Theorem 5, the number of iterations needed in APG to achieve ϵ -solution is $O(\frac{\|\boldsymbol{\omega}^0 - \boldsymbol{\omega}^*\|}{\sqrt{\epsilon}})$. Hence a good initial guess of $\boldsymbol{\omega}^0$ can greatly save the computational cost. When a new active constraint is added in the active constraint set, we can use the optimal solution of the last iteration (denoted by $[\boldsymbol{\omega}_1^{*'}, \dots, \boldsymbol{\omega}_{T-1}^{*'}]$) as an initial guess to do the next cutting plane iteration, which can greatly improve the convergence speed. In other words, at the T th iteration, we use $\boldsymbol{\omega}^{-1} = \boldsymbol{\omega}^0 = [\boldsymbol{\omega}_1^{*'}, \dots, \boldsymbol{\omega}_{T-1}^{*'}, \boldsymbol{\omega}_T']$ as the starting point, where $\boldsymbol{\omega}_T = \mathbf{0}$.

6. Experiments on Large-scale Linear Feature Selection

In this section, we show the performance of the proposed methods on large-scale linear feature selection tasks on a collection of synthetic datasets and real-world datasets. The experiments on group selection as well as nonlinear feature selection will be given in Section 7.

As our focus is on large-scale and very high-dimensional problems, some previously mentioned methods, such as NMMKL and QCQP-SSVM cannot be applied due to their high computation cost. And we also do not consider the SVM-RFE method due to its monotonicity and sub-optimality to feature selections. For complete comparisons with QCQP-SSVM, NMMKL and SVM-RFE, interested readers can refer to the conference version (Tan et al., 2010). Besides these, we will also present an independent experiment to demonstrate the advantages of the proposed methods over traditional ℓ_1 -regularized methods on second-order feature selections with polynomial feature mappings. All the referred methods are implemented in C++ and the source codes are available at <http://c2inet.sce.ntu.edu.sg/Mingkui/robust-FGM.rar>.

6.1 Datasets and Experimental Setup

Seven large-scale and high dimensional datasets are used for the study of linear feature selections. General information of these datasets including the dimension m , the number of training points (n_{train}), the number of testing points (n_{test}) and the average nonzero features per instance are listed in Table 1. For `epsilon`, `Arxiv astro-ph`, `rcv1.binary` and `kddb` datasets, they have already been split into training set and testing set. For `aut-avn`, `real-sim` and `news20.binary` datasets, we manually split them into training and testing sets, as shown in Table 1. For all datasets, we do not perform any data normalization and hence the scaling factor is fixed by $\boldsymbol{\lambda} = \mathbf{1}$. All experiments are performed on a 2.27GHZ Intel(R)Core(TM) 4 DUO CPU running windows sever 2003 with 24.0GB main memory.

Comparisons are conducted between FGM and ℓ_1 -regularized methods, including ℓ_1 -SVM and ℓ_1 -LR. For FGM, we study FGM with SimpleMKL solver (denoted by MKL-FGM⁴) (Tan et al., 2010), FGM with accelerated proximal gradient solver using square hinge loss (denoted by PROX-FGM) and logistic loss (denoted by PROX-SLR). For simplicity, when we use the term FGM, hereafter it includes all FGM-based methods, including MKL-FGM, PROX-FGM and PROX-SLR. Recently, many efficient solvers for ℓ_1 -SVM and ℓ_1 -LR have been developed for large-scale problems. Typical methods include the interior point method (IPM), fast iterative shrinkage-threshold algorithm (FISTA), block coordinate descent (BCD), Lassplore method, generalized linear model with elastic net (GLMNET) and so on (Liu and Ye, 2010; Yuan et al., 2010, 2011). Among various implementations, the LIBLinear solver that uses the coordinate descent method with improved line search has demonstrated state-of-the-art performance in terms of training efficiency and generalization performance (Yuan et al., 2010). Particularly, by taking the advantages of sparsity in data, LIBLinear can achieve even faster convergence speed on sparse datasets (Yuan et al., 2010, 2011). Therefore, we only include the latest LIBLinear solver for ℓ_1 -SVM and ℓ_1 -LR⁵ for comparison, which are denoted by l1-SVM and l1-LR, respectively. Besides, we also use the coordinate descent solver in LIBLinear to train a standard SVM and LR classifier with all features as the baselines, denoted by CD-SVM and CD-LR, respectively. We use the default stopping criterion in LIBLinear for all methods, while the parameter settings for FGM will be discussed later. As a complement, the comparison of FGM with FISTA and BCD can be seen in Section 7.

Dataset	m	n_{train}	n_{test}	# nonzeros per instance	Parameter range for C	
					l1-SVM	l1-LR
epsilon	2,000	400,000	100,000	2,000	[0.0005, 0.01]	[0.001, 0.1]
aut-avn	20,707	40,000	22,581	50	[0.005, 0.08]	[0.01, 0.4]
real-sim	20,958	32,309	40,000	52	[0.005, 0.3]	[0.005, 0.06]
rcv1	47,236	677,399	20,242	74	[0.0001, 0.004]	[0.00005, 0.002]
astro-ph	99,757	62,369	32,487	77	[0.005, 0.06]	[0.02, 0.3]
news20	1,355,191	9,996	10,000	359	[0.005, 0.3]	[0.05, 1.50]
kddb	29,890,095	19,264,097	748,401	29	[0.000005, 0.0003]	[0.000003, 0.0001]

Table 1: The statistics of datasets used in the experiments. Among them, **epsilon**, **real-sim**, **rcv1.binary**, **news20.binary** and **kddb** are downloaded from LIBSVM website: <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>. **aut-avn** can be downloaded from <http://vikas.sindhwani.org/svmlin.html> and Arxiv **astro-ph** is from (Joachims, 2006).

4. The code can be downloaded from: <http://c2inet.sce.ntu.edu.sg/Mingkui/FGM.htm>.

5. The code can be downloaded from: <http://www.csie.ntu.edu.tw/~cjlin/liblinear/>.

6.2 Experiments on Synthetic Dataset

6.2.1 SYNTHETIC DATASET GENERATION

Before we evaluate the performance of feature selection methods on real-world datasets, we first thoroughly analyze their performances on three synthetic experiments. Specifically, three types of data matrices with increasing size, namely, $\mathbf{X}_1 \in \mathbb{R}^{4,096 \times 4,096}$, $\mathbf{X}_2 \in \mathbb{R}^{4,096 \times 65,536}$ and $\mathbf{X}_3 \in \mathbb{R}^{8,192 \times 100,000}$ are generated as datasets in 4,096, 65,536 and 100,000 dimensions, respectively. Each entry of \mathbf{X} is sampled from the *i.i.d.* Gaussian distribution $\mathcal{N}(0, 1)$. To produce the labels \mathbf{y} , we first generate a sparse weight vector \mathbf{w}_0 as the ground truth weights to the features. Among all m dimensions in \mathbf{w}_0 , only 400 of them, which are sampled from the Uniform distribution $\mathcal{U}(0, 1)$, are the ground truth informative features; and the rests are set to zeros. By sorting the non-zero entries of \mathbf{w}_0 in ascending order, the weight distribution of the top 400 features is shown in Figure 1(a), denoted by **Type-I** weighting. To make a thorough comparison, we also generate another two types of weights by using $\mathbf{w}_1 = \mathbf{w}_0^{0.3}$ and $\mathbf{w}_2 = \mathbf{w}_0^3$, which are denoted by **Type-II** and **Type-III** weightings in Figure 1(a). Compared with **Type-I** weighting, the decrement of the non-zero entries of \mathbf{w}_1 for **Type-II** weighting is relatively more stable; while for **Type-III** weighting, the non-zero entries of \mathbf{w}_2 have a much sharper decrease. After generating the ground-truth weight vector \mathbf{w} , the output labels are generated by $\mathbf{y} = \text{sign}(\mathbf{X}\mathbf{w})$. For the two larger datasets, only the **Type-I** weighting will be studied. Following the same way, we can generate the testing datasets \mathbf{X}_{test} and the output labels $\mathbf{y}_{test} = \text{sign}(\mathbf{X}_{test}\mathbf{w})$. The number of testing points is set to 4,096 for all cases.

6.2.2 EXPERIMENTS ON SMALL SYNTHETIC DATASET

The simulation on the smallest synthetic dataset, namely, $\mathbf{X}_1 \in \mathbb{R}^{4,096 \times 4,096}$, includes the following several aspects. First of all, we show the convergence behavior of the proposed methods on this dataset. As PROX-SLR has the similar performance as PROX-FGM, we only show the convergence behavior of PROX-FGM on **Type-I** weighting. For the parameter settings, because we know in advance that there are 400 informative features, we test PROX-FGM with $B = 10, 50, 100, 200, 500$. For simplicity, we set $C = 10$ in experiments. In addition, the stopping criterion for inner APG algorithm is set to

$$\left| \frac{f_{APG}^{k-1} - f_{APG}^k}{f_{APG}^{k-1}} \right| \leq \epsilon_{APG} = 1.0 \times 10^{-4}, \quad (43)$$

where f_{APG}^k denotes the objective value in APG. As in each outer iteration, several APG iterations are needed to solve the QCQP subproblem, we stop the outer cutting plane algorithm after 300 total inner APG iterations are achieved.

The objective function value evolutions within 300 APG iterations are shown as in Figure 1(b), where we can observe that the function values strictly decrease and have a sharp decline at each cutting plane iteration where an new active constraint is included. Moreover, in each cutting plane iteration, the APG algorithm shows superlinear convergence behavior. However, the function value does not significantly increase after several APG iterations at each cutting plane iteration, which indicates that the stopping criterion (43) is sufficient for the inner APG algorithm.

(a) Types of the ground truth in- (b) Function values over all APG (c) Function difference over cut-
formative features iterations on **Type-I** ting plane iterations on **Type-I**

Figure 1: Experiments on $\mathbf{X} \in \mathbb{R}^{4,096 \times 4,096}$. (a): For **Type-I** weighting, the ground truth \mathbf{w}_0 of informative features is sampled from Uniform distribution $\mathcal{U}(0, 1)$; while **Type-II** and **Type-III** are generated via $\mathbf{w}_1 = \mathbf{w}_0^{0.3}$ and $\mathbf{w}_2 = \mathbf{w}_0^3$, respectively. The curve in the figure is obtained by sorting the features in ascending order according to their weights. (b) & (c): The APG iteration counts the total number of evolved APG iterations while the Outer iteration counts the number of outer cutting plane iterations.

To better understand the convergence behavior of the cutting plane algorithm used in FGM, In Figure 1(c), we show the function value difference between the successive cutting plane iterations, where we use the stopping criterion for the outer cutting plane algorithm as in Figure 1(c) or the relative function difference $\left| \frac{f^{T-1} - f^T}{f^{T-1}} \right| \leq \epsilon_c = 1.0 \times 10^{-2}$. Figure 1(c) shows that the function value difference have a monotonically decrease. An important observation from Figure 1(b) and 1(c) is that, with a fixed C , PROX-FGM converges faster with larger B and needs more outer cutting plane iterations for smaller B . This fact can be easily understood for that we have 400 ground truth informative features. Then a small B denotes a very loose guess of the number of ground truth features.

Notice that, in some cases where we just need to find a small subset of the informative features, a smaller B is desirable and more meaningful, which is a good start point of most feature selection tasks. In such cases, given the guess of B , we can terminate the algorithm after T outer cutting plane iterations. And the number of selected features will be not more than TB . In our subsequent experiments, except for specification, we terminate PROX-FGM and PROX-SLR, when either $\left| \frac{f^{T-1} - f^T}{f^{T-1}} \right| \leq \epsilon_c = 1.0 \times 10^{-2}$ or the maximum number of outer iterations T is achieved. By default, we set the maximum outer iterations to 15.

In the second experiment, we report the prediction accuracy of the referred methods on the synthetic dataset w.r.t the number of selected features. For FGM, we fix $C = 10$ and vary $B \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 24, 25, 26, 28, 30, 32, 34, 36, 38, 40\}$ to obtain different number of selected features. For l1-SVM and l1-LR, we need to tune the trade-off parameter C to obtain different sparsity levels of solutions. Specifically, we vary $C \in [0.001, 0.007]$ for l1-SVM and $C \in [0.005, 0.040]$ for l1-LR to achieve different desired sparse levels of

(a) Type I (b) Type II (c) Type III

Figure 2: Testing accuracy versus the selected features on three types of synthetic datasets ($n = 4,096, m = 4,096$), where CD-SVM and CD-LR denote the result by using standard SVM and LR with all features.

solutions⁶. The testing accuracy w.r.t the number of selected features on the three types of weightings are reported in Figure 2. From Figure 2, we can observe that with suitable number of selected features, feature selection can be significant better than CD-SVM and CD-LR with all features in terms of prediction accuracy. And the performance will decline if too many features are selected. These facts verify the importance of feature selections.

In addition, FGM can always be better than ℓ_1 -methods in terms of prediction accuracy on all three types of weights. Notice that here we carefully change C for ℓ_1 methods to achieve different sparsity, hence the results cannot be further improved via parameter tunings. In short, for FGM, we can fix C and tune either B or T to achieve different desired sparsity; while traditional ℓ_1 -methods require to vary C very prudently for trading the desired sparsity with the fitness of the decision function. Obviously, tuning C is not an intuitive way for controlling the number of features. Herein a natural question arises: why does FGM show better prediction accuracy under the same number of selected features?

To better understand this phenomenon, we first report the number of recovered ground truth features of the selected features in Figure 3. It is apparent that we cannot recover all ground truth features for classification problems because of the round errors brought by $\mathbf{y} = \text{sign}(\mathbf{X}\mathbf{w})$. Moreover, it is obvious that for a fixed feature subset, the larger number of recovered ground truth features indicates better ability and smaller scaling bias for feature selection. Let m_s be the number of selected features. From Figure 3, we can observe that, with the same number of selected features, FGM based methods can always include more ground truth features than ℓ_1 -methods when $m_s \geq 200$ for all three types of features, which coincide with the improved prediction performances of FGM in Figure 2. Hence it is not surprising that FGM obtain better prediction accuracy under the same sparsity.

Moreover, when $m_s \leq 200$ for Type-I and Type-II and $m_s \leq 100$ for Type-III, although FGM selects the close number of informative features to ℓ_1 -methods, it can still obtain better prediction accuracy. One possible reason is that the features selected by FGM are more informative than ℓ_1 -methods, which can be brought by the scaling bias of the ℓ_1 regularization. Such bias problem has been studied for ℓ_1 -regularized linear re-

6. For l1-SVM and l1-LR, to obtain desired number of selected features, the range of C has to vary a lot for different problems, hereafter we do not report the specific C values but give its range for the presentation issues.

(a) Type I (b) Type II (c) Type III

Figure 3: Number of recovered features over all selected feature ($n = 4,096, m = 4,096$).

gressions (Figueiredo et al., 2007). To alleviate the bias problem, a de-bias procedure is suggested after the ℓ_1 optimization with the selected features (Figueiredo et al., 2007). That is to say, we can assume that the selected features are informative and then apply a de-bias procedure to reduce the bias. For classification problem, different from the linear case, we can apply a standard SVM or LR to train an unbiased model with relatively large C parameter. Here only the squared hinge loss (SVM) is studied. The results are reported in Figure 4, where l1-SVM-unbias and PROX-SVM-unbias denote the de-biased results of l1-SVM and PROX-FGM by using standard linear SVM with $C = 20$, respectively. Then, if there is no bias, both FGM and ℓ_1 -methods should obtain close results to their counterparts, namely, l1-SVM-unbias and PROX-SVM-unbias, respectively. From Figure 4, l1-SVM-unbias is generally better than l1-SVM, which indicates that the bias problem for ℓ_1 -SVM really exists. On the contrary, PROX-FGM can get close or even better results than PROX-SVM-unbias, which demonstrates that FGM indeed can alleviate the scaling bias problem. What's more, l1-SVM-unbias is a bit worse than PROX-SVM. It further indicates that the selected features of ℓ_1 -SVM may not be so informative as PROX-SVM due to the bias problem.

(a) Type I (b) Type II (c) Type III

Figure 4: De-bias effect of FGM on synthetic dataset ($n = 4,096, m = 4,096$).

An interesting comparison is among the three types of features in terms of accuracy. As from Figure 2 and 3, the Type-III features with the least recovered features produces the best prediction accuracy while the Type-II features produces the worst prediction accuracy with the most recovered features. This phenomenon can be easily understood from Figure 1(a), where most of the generated features are with relatively large weights and informative for Type-II while only half of the generated features are significantly informative with relatively large weights for Type-III. The features with small weights can be neglected

without much loss of the generalization performance. As a result, **Type-III** features should be the easiest case to solve.

(a) **Type I** (b) **Type II** (c) **Type III**
Figure 5: Training time on synthetic datasets ($n = 4,096, m = 4,096$).

The final comparison issue is regarding the efficiency of various methods. The training time of the refereed methods on $\mathbf{X} \in \mathbb{R}^{4,096 \times 4,096}$ are listed in Figure 5. From Figure 5, we can observe that **PROX-FGM** and **PROX-SLR** are comparable with ℓ_1 -methods, standard SVM (CD-SVM) and LR (CD-LR). Particularly, **PROX-FGM** and **PROX-SLR** are much faster than **MKL-FGM** which adopts the SimpleMKL as the QCQP solver. Specifically, **PROX-FGM** and **PROX-SLR** can gain up to 1,000 times faster than **MKL-FGM**. The reason for this fact is that SimpleMKL has to train a SVM QP problem to obtain the subgradient, and needs thousands of independent SVM trainings to converge. However, in **PROX-FGM** and **PROX-SLR**, APG is adopted to solve the QCQP problem in one shot and can significantly improve the training efficiency. Besides the training efficiency, **PROX-FGM** and **PROX-SLR** can also obtain slightly better results than **MKL-FGM** in terms of the prediction accuracy. This is because that the accuracy of the subgradient in SimpleMKL obtained by SVM solver largely depends on the accuracy of the SVM optimization. Unfortunately, a less accurate solution can improve the convergence speed but will significantly degrades the accuracy of the QCQP solution. Hence SimpleMKL solver may not be suitable for large-scale problems.

(a) Testing accuracy (b) Recovered features (c) Training time

Figure 6: Performance comparison on large-scale synthetic dataset ($n = 4096, m = 65536$).

6.2.3 EXPERIMENTS ON LARGE-SCALE SYNTHETIC DATASET

As shown in Figure 5, on the relatively small problems, **PROX-FGM** and **PROX-SLR** are comparable in terms of efficiency with the ℓ_1 -methods solved by LIBLinear. To further

(a) Testing accuracy (b) Training time

Figure 7: Performance comparison on synthetic dataset ($n = 8,192$, $m = 100,000$) with different data densities in $\{0.005, 0.01, 0.05, 0.1, 0.5, 1\}$.

demonstrate the scalability of various methods, we perform a comparison on the two much larger synthetic datasets. In this experiment, we only include the results of the **Type-I** weighting. In addition, we do not include the results of MKL-FGM for its inefficiency. Two different experiments are conducted.

In the first experiment, on dataset $\mathbf{X}_2 \in \mathbb{R}^{4,096 \times 65,536}$, the testing accuracy, training time and the number of recovered ground truth features within the selected features are reported in Figure 6(a), 6(b) and 6(c), respectively. Here, we keep the same parameter settings for FGM as above and vary $C \in [0.001, 0.004]$ for l1-SVM and $C \in [0.005, 0.015]$ for l1-LR. From Figure 6(a) and 6(b), we can observe that both PROX-FGM and PROX-SLR outperform their counterpart with ℓ_1 -methods in terms of prediction accuracy and the number of recovered ground truth features. More importantly, from Figure 6(c), PROX-FGM and PROX-SLR have better training efficiency than ℓ_1 -methods with coordinate descent solvers. This superiority will be even more significant when dealing with even larger and higher dimensional problems. Notice that, although both the coordinate descent methods for ℓ_1 and FGM scale $O(mn)$, FGM can be much more efficient in the sense that it only requires tens of times to scan the whole dataset, and it solves a much smaller optimization problem with the complexity $O(nTB)$.

In LIBLinear, the efficiency of ℓ_1 methods solver has been much improved by taking the advantage of the sparsity of the datasets. In the second experiment, we test the sensitivity of referred methods on dataset $\mathbf{X}_3 \in \mathbb{R}^{8,192 \times 100,000}$ with different data densities. Specifically, we generate datasets with different densities by sampling the entries from \mathbf{X}_3 with sampling rate $\{0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ and test the influence of the data sparsity towards different learning algorithms. Here only the square hinge loss (SVM) is studied. For sake of brevity, we only report the **best accuracy** obtained among all parameters and the corresponding training time of l1-SVM and PROX-FGM in Figure 7. For parameter settings, we keep the default experimental settings for PROX-FGM and watchfully vary $C \in [0.008, 5]$ for l1-SVM. From Figure 7(a), we can observe that, for different data densities, PROX-FGM can always outperform l1-SVM in terms of the **best accuracy**. From Figure 7(b), we can see that l1-SVM has comparable efficiency with PROX-FGM on low data density cases but much worse efficiency than PROX-FGM when the dataset are relative denser. This facts verify that FGM has good scalability over increasing data densities while the efficiency of l1-SVM with coordinate solver highly depends on the dataset density.

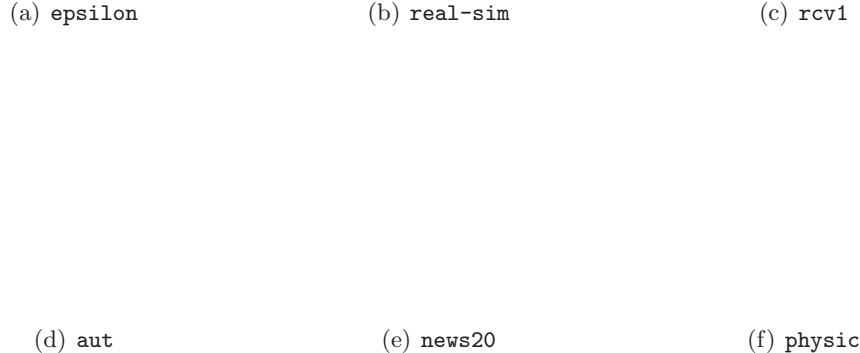


Figure 8: Testing accuracy on various datasets

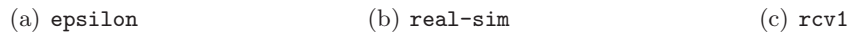


Figure 9: De-bias effect of FGM on real-world dataset.

6.3 Experiments on Real-world Datasets

In this subsection, we are ready to verify the performance of FGM on several well-known benchmark datasets listed in Table 1. Unlike in the synthetic experiments, we have no idea about the number of ground truth informative features in advance any more for the real-world datasets. Considering that feature selection task prefers to selecting a small number of features in real applications, we set the parameter B for FGM based methods in the range of $\{2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 24, 26, 30, 32, 35, 38, 40, 42, 45, 48, 50, 55, 60\}$ with fixed $C = 10$ while the range of C for ℓ_1 -methods varies for different problems, which is listed in Table 1.

In the first experiment, we reported the testing accuracy and the training time against the number of selected features of the first six datasets in Figure 8 and Figure 10 respectively. We do not include the results of MKL-FGM on `epsilon` dataset due to its inefficiency. From Figure 8, we can see that on all datasets, FGM (including PROX-FGM, PROX-SLR and MKL-FGM) can obtain superior or comparable performance than ℓ_1 -methods in terms of

prediction accuracy within the 300 selected features. Specifically, FGM shows much better prediction accuracy on `epsilon`, `real-sim`, `rcv1.binary`, `Arxiv astro-ph` and `news20`. On `aut-avn`, although there is no significant superiority over ℓ_1 -methods on the whole range, FGM shows much better performance in the magnified sub-figures.

Again, to explain why FGM can perform better in terms of prediction accuracy, similar in the synthetic experiments, we did an independent experiment to show the bias problem of ℓ_1 -methods. Here we only studied the square hinge loss. We train the unbiased model via using the standard SVM with the selected features obtained by l1-SVM and PROX-FGM, denoted by PROX-FGM-unbias and l1-SVM-unbias, respectively. If there is no bias, the prediction accuracy of PROX-FGM-unbias and l1-SVM-unbias should be close to their counterparts. However, from Figure 9, we can observe that l1-SVM-unbias shows much better results than l1-SVM, which indicates that the bias problem really happens for l1-SVM. On the contrary, PROX-FGM obtains very close to or even better results than PROX-FGM-unbias on three datasets. Another factor that possibly contributes to the good performance of FGM is that the features selected by FGM can be more informative than those obtained by ℓ_1 -methods. Notice that l1-SVM-unbias can be considered as an unbiased remedy to l1-SVM. In addition, on all three datasets, FGM shows better results than l1-SVM-unbias, which verifies the above arguments.

Figure 10 shows the training time of various methods, from which we can observe that PROX-FGM and PROX-SLR shows comparable training efficiency over ℓ_1 -methods. Particularly, on `epsilon` dataset (which is in dense format from Table 1), PROX-FGM and PROX-SLR is much faster than the ℓ_1 counterparts. Notice that other datasets are very sparse and the coordinate descent methods can be very fast by taking the advantage of data sparsity. With these facts, FGM is expected to obtain better efficiency on dense datasets of large-scale and very high-dimensional problems.

A final observation is that on all datasets, the standard SVM (denoted by CD-SVM) and LR (denoted by CD-LR) with all features can obtain a bit higher prediction accuracy. One possible reason accounting for this is that 300 features may be not enough to cover all the informative features. And as from the trend of the curve, we can see that the performances can be improved by including more features. In this sense, we are aware of that the feature selection does not necessary improve the generalization performance on all problems. In other words, whether or not the feature selection can improve the generalization performance is data-dependent. For example, the following experiments presents a example that feature selection can really help to improve the generalization performance on `kddb` dataset.

From Table 1, `kddb` is very large-scale in training examples and extremely high dimensional in features. Thus we will not include the results of MKL-FGM for its inefficiency on dealing with large problems. To thoroughly study it, we first use the first 1×10^5 and 1×10^6 examples from the training data to test how the performance varies w.r.t the number of training points. We follow the same experimental settings above and report the testing accuracy and training time in Figure 11 and 12, respectively. From Figure 11, on all the three datasets of different scales, the feature selection can much improve the generalization performance over the standard SVM and LR that use all features. More importantly, FGM shows better or comparable performance over the ℓ_1 counterparts. Regarding the issue of training efficiency, interestingly, we find that CD-SVM and CD-LR needs much more training time, which indicates that with feature selection, the training efficiency can be possibly

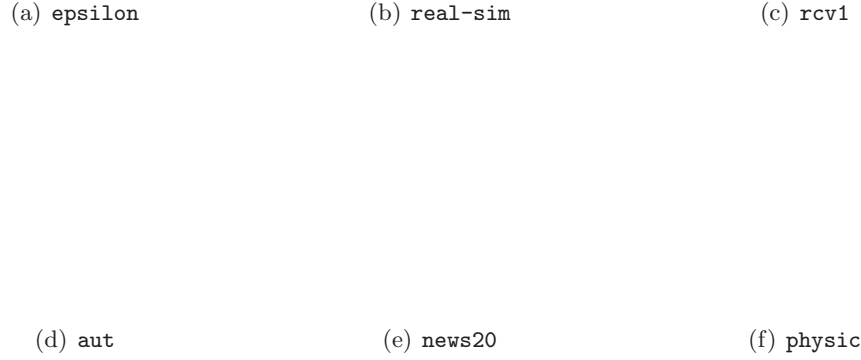


Figure 10: Training time on various datasets.

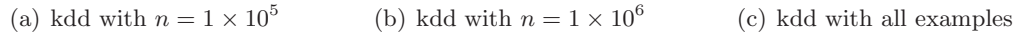


Figure 11: Testing accuracy on kddb.

improved. A final observation in Figure 11 is that with more and more training examples, the gap between ℓ_1 -methods and FGM becomes narrowed.

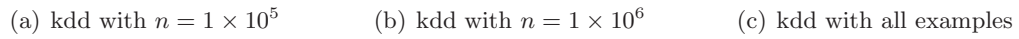


Figure 12: Training time on kddb.

Dataset	m	m_{Poly}	n_{train}	n_{test}	γ
mnist38	784	$O(10^5)$	40,000	22,581	4.0
real-sim	20,958	$O(10^8)$	32,309	40,000	8.0
kddb	4,590,807	$O(10^{14})$	1000,000	748,401	4.0

Table 2: Details of the datasets with polynomial feature mappings.

6.4 Second-order Nonlinear Feature Selection with Polynomial Feature Expansions

For general kernels, the associated feature selection can be very hard. However, if the explicit feature expansions of input features for the kernels can be available in advance, such as the polynomial kernels and string kernels (Chang et al., 2010; Sonnenburg et al., 2006), the feature selection problem can be solved via the linear techniques. Taking the polynomial kernel, namely, $k(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i' \mathbf{x}_j + r)^v$, as an example, when the degree parameter $v = 2$, we can obtain its explicit feature map as: $\phi(\mathbf{x}) = [r, \sqrt{2\gamma r}x_1, \dots, \sqrt{2\gamma r}x_m, \gamma x_1^2, \dots, \gamma x_m^2, \sqrt{2\gamma}x_1x_2, \dots, \sqrt{2\gamma}x_{m-1}x_m]$. Compared with the linear case, this second-order feature mapping can obtain the feature pair dependencies. A critical problem for such kind of explicit feature mappings is their extremely high dimensions in the expanded feature space. For polynomial kernel, the dimension of the feature mapping will exponentially increase with v . When the order $v = 2$, the dimension is $(m+2)(m+1)/2$. Typically, when $m = 10^6$, the expanded dimension will be around 10^{12} . In such case, we need around 1,000 GB to store a full weight vector \mathbf{w} , making it intractable (Chang et al., 2010). However, thanks to the scheme of FGM, as the discussion in Section 5.3, the computational issues can be addressed because we only need to keep the score and the feature copy of the TB features (or groups) in the main memory.

The second order ($v = 2$) feature selection with polynomial expansions can learn the combination of any two features, which is particularly important for discovering the feature interactions. In this experiment, we will show the efficiency of FGM with polynomial feature mappings on two medium dimensional datasets and one very-high dimensional problems. The details of the datasets are shown in Table 2, m_{Poly} denotes the dimension of the polynomial mappings and γ is the polynomial kernel parameter. The **mnist38** two-class dataset consists of the digital images of 3 and 8 from **mnist**, which can be downloaded from LIBSVM website. To impose greater importance on the nonlinear features, we set $\gamma = 8$ for **real-sim** dataset. For **kddb** dataset, we only use the first $n = 1 \times 10^6$ instances. Again, only the squared hinge loss is studied. In the experiment, we vary the B for FGM and C for l1-SVM with polynomial mappings (denoted by l1-PSVM) to obtain different number of features.

The corresponding testing accuracy and training time are shown in Figure 13 and 14, respectively. Notice that, l1-PSVM cannot work on the **kddb** dataset because of the extremely high-dimensions (as shown in Table 2), and our computer cannot afford enough memory for storing the decision vector \mathbf{w} either. Conversely, with the FGM scheme, this problem can be efficiently solved. As From Figure 13(c), PROX-PFGM and PROX-PSLR can finish the training within 1000 seconds. l1-PSVM can work on the medium dimensional problems. However, PROX-PFGM shows much better training efficiency over l1-PSVM. Finally, we can observe from Figure 14 that, with polynomial mappings, the prediction accuracy on

`mnist38` can be much improved over the linear methods, which verifies the importance of the second-order feature selections. However, on `real-sim` and `kddb` dataset, the performance does not show any improvements. The reason is that the latter two datasets tend to be linearly separable. And if we impose more importance on the second-order feature pairs, the performance may even decrease.

(a) `mnist38` (b) `real-sim` (c) `kddb`

Figure 13: Training time on various datasets with polynomial mappings. The result for l1-PSVM on `kddb` is absent for computational issues.

(a) `mnist38` (b) `real-sim` (c) `kddb`

Figure 14: Testing accuracy on various datasets with polynomial mappings. The result for l1-PSVM on `kddb` is absent for computational issues.

7. Experiments on Large-scale Group Feature Selection

In this experiment, we will verify the performance of the feature generating scheme on group selections on two synthetic datasets and two real-world datasets. From the linear feature selection experiments, we can see that the performances of the hinge loss and logistic loss are quite similar. In addition, when doing group selection for classification problems, the logistic loss is more often studied (Roth and Fischer, 2008; Liu and Ye, 2010). Hence, we only studied the logistic loss. Finally, although the feature generating scheme can be easily extended for dealing with overlapping groups and tree-structured groups, we only studied the non-overlapping groups in this experiment.

7.1 Experimental Settings

Three recently developed methods, namely, the accelerated proximal gradient descent (denoted by FISTA) (Liu and Ye, 2010; Jenatton et al., 2011; Bach et al., 2011), block coordinate descent method (denoted by BCD) (Qin et al., 2010.) and active set method (denoted by ACTIVE) (Bach, 2009; Roth and Fischer, 2008) have been adopted as the baseline methods. Among them, FISTA uses the accelerated proximal gradient descent have been thoroughly studied by several researchers (Liu and Ye, 2010; Jenatton et al., 2011; Bach et al., 2011). Here we use the implementation of SLEP package⁷, where an improved line search is used. We implement by ourselves the block coordinate descent method proposed by Qin et al. (2010.), where each sub-problem is solved by accelerated proximal gradient methods. Finally, we adopt the FISTA solver to implement the ACTIVE method, which has a similar structure to FGM based methods. Different from FGM, FISTA, BCD and ACTIVE solves a standard group lasso problem (Liu and Ye, 2010). All the referred methods are implemented in MATLAB for fair comparison. All experiments are performed on a 2.27GHZ Intel(R)Core(TM) 4 DUO CPU running windows sever 2003 with 24.0GB main memory.

7.2 Experiments on Synthetic Datasets

Two synthetic experiments was conducted to verify the performance of various methods. Specifically, we generate two random matrices $\mathbf{X} \in \mathbb{R}^{n \times m}$ of different dimensions ($m = 20,000$ and $m = 400,000$, respectively) as the datasets. Each element of \mathbf{X} follows *i.i.d.* Gaussian distribution $\mathcal{N}(0,1)$. For both datasets, the number of examples is set to $n = 4,096$. In addition, we naturally group the features into groups of equal size of 10, resulting in 2,000 and 40,000 groups, respectively. Among all the groups, we randomly select 100 of them as ground truth informative groups and generate a sparse ground truth vector \mathbf{w} , where only the entries of the selected features are nonzeros sampled from *i.i.d.* Gaussian distribution $\mathcal{N}(0,1)$. Finally, we produce the output labels by $\mathbf{y} = \text{sign}(\mathbf{X}\mathbf{w})$. We can generate the testing dataset by using the same way, where 2000 testing points are generated.

To fairly compare with other baseline methods, we make a slightly different setting of FGM from the experiments in Section 6. That is, we set a large number T and vary C regarding each of the predefined $B \in \{2, 5, 8, 10\}$ to obtain different number of groups. Notice that the tradeoff parameter λ in SLEP is in the range of $[0, 1]$, and a larger lambda will lead more sparse results (Liu and Ye, 2010). Hence we set C in $\{0.002, 0.006, 0.010, 0.016, 0.020, 0.060, 0.100, 0.160, 0.200, 0.300, 0.400, 0.500, 0.600, 0.700\}$ for FISTA and ACTIVE. And we set C for BCD in the range $\{0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.011, 0.013, 0.015, 0.018, 0.02, 0.023, 0.025, 0.028, 0.03, 0.05, 0.1\}$. The testing accuracy, training time and the number of recovered ground truth groups are reported in Figure 15(a), 15(b) and 15(c), respectively. Here only the results within 150 features are included because we have only 100 informative ground truth groups. From Figure 15, we can observe that, FGM can obtain better prediction accuracy than FISTA and BCD methods. Interestingly, the ACTIVE set method also shows better performance in terms of prediction accuracy over FISTA and BCD. Similar in the linear feature selection experiment, it is because FGM can alleviate the bias problem, which will be further discussed in Section 8. Notice that ACTIVE is directly adapted from FISTA but adopts the **generating** scheme by including one

7. <http://www.public.asu.edu/~jye02/Software/SLEP/index.htm>.

feature group at each time. Its superior generalization performance further demonstrates the effectiveness of the feature generating scheme. However, as from Figure 15(c), **ACTIVE** is limited for its computational cost on large-scale problems, where the ground informative groups is relatively very large or the number of patterns is very large. From Figure 15(c), FGM is much more efficient than FISTA and BCD with a relatively large B . Because the ground truth number of informative groups is 100, a larger B that is close to 100 can converge faster. In addition, with a fixed B , the number of selected will increase when C becomes large. This is because, when we set a large C , we impose more importance on the training errors. Hence, FGM requires more features to obtain a sufficient decrease of errors on the training dataset. However, as in Section 6, we can also control the number of groups by stopping the cutting plane algorithm with predefined iterations.

(a) Testing accuracy (b) Recovered features (c) Training time

Figure 15: Results on synthetic dataset with $m = 2 \times 10^5$.

(a) Testing accuracy (b) Recovered features (c) Training time

Figure 16: Results on synthetic dataset with $m = 4 \times 10^6$.

7.3 Experiments Real-world Datasets

In this experiment, we will demonstrate the effectiveness of the proposed methods on group selections on two real-world datasets. In real-applications, the group prior of features can be extracted in different ways. The generation of group prior, however, is beyond the scope of this paper. Notice that the explicit feature mapping can be a natural way to generate groups (Wu et al., 2010; Vedaldi and Zisserman, 2010). In this paper, we produce the feature groups by using explicit kernel feature mappings, where each feature will be expanded by a group of features. As from the referred papers, such expansion can obtain very good performance in some computer vision applications. More importantly, the authors pointed out the explicit

Dataset	m	Storage of training set (GB)				Storage of testing set(GB)			
		n_{train}	Linear	ADD	HIK	n_{test}	Linear	ADD	HIK
aut	20,707	40, 000	0.027	0.320	0.408	22,581	0.016	0.191	0.269
rcv1	47,236	677,399	0.727	8.29	9.700	20,242	0.022	0.256	0.455

Table 3: Details of the datasets for group feature expansion of HIK kernel and Additive kernel. For HIK kernel expansion, each feature is represented with a group of 100 features while for Additive kernel expansion, each feature is represented with a group of 11 features.

features can potentially work well in many machine learning problems beyond computer vision applications (Wu et al., 2010).

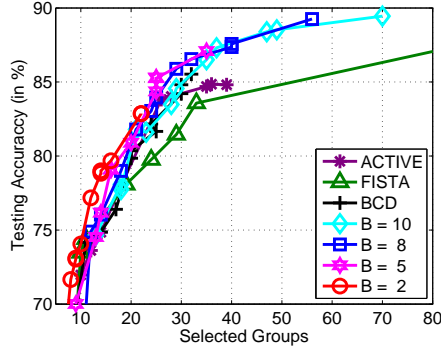
Here, only the HIK kernel and the additive Gaussian kernel are used for our studies. The details and the code of the feature expansion can be referred in (Wu et al., 2010) and (Vedaldi and Zisserman, 2010), respectively. In our experiments, we use **aut** and **rcv1** for case studies. For fair comparison, we pre-generate all features before we do group selection. Some information regarding the feature expansion is listed Table 3. From Table 3, we can see that after feature expansion, the storage requirement of the dataset will heavily increase. Actually, this expansion cannot be explicitly performed when dealing with large-scale and very high dimensional problems for the unbearable storage requirements. Accordingly, FISTA and BCD, which requires the knowledge of the explicit presentation of the dataset, cannot work anymore in such case ⁸. On the contrary, with the feature generating scheme, FGM and ACTIVE method can successfully works on such situations because only much reduced problems need to be solved. Figure 17 and 18 reported the testing accuracy and training time of various methods, respectively. From these two figures, we can easily observe the superior performance of FGM and the active set method over FISTA and BCD in terms of prediction accuracy. However, FGM can gain much better efficiency than the active set method.

8. Related Studies and Discussions

FGM highly relates to multiple kernel learning (MKL) algorithms. Specifically, the sub-problem of FGM can be solved via the SimpleMKL algorithm. However, FGM differs from MKL in the following aspects:

- FGM starts to do linear feature selection via an adaptive feature scaling problem. With a predefined scalar B , FGM automatically generates a series of feature groups of size B via the worst-case analysis, where each group of features can form as a kernel.
- It is also natural to extend FGM to do multiple kernel learning problems where the kernels to be learned can be organized as additive kernels. Under this condition, the cutting plane algorithm includes a group of B kernels each iteration and the

8. For HIK kernel, the expansion can be done in an implicit way according to (Wu et al., 2010). However, it is not considered in this experiment.



(a) aut-ADD

(b) rcv1-ADD

(c) aut-HIK

(d) rcv1-HIK

Figure 17: Testing accuracy on group selections. The groups are generated by either HIK or additive feature mappings. The result of BCD for aut-HIK cannot be available for the heavy computational cost.

global convergence can be guaranteed. Hence FGM is very suitable for MKL learning problems with a large number of kernels.

- In FGM, some techniques are proposed to accelerate the convergence. Particularly, for MKL learning problems, we can apply explicit feature mappings for the kernels, which is important for the scalability of the large-scale problems with extremely large number of features or kernels.

To address the challenges brought by the extremely large number of features or kernels, the active method has been used to tackle the huge search space problems (Bach, 2009; Roth and Fischer, 2008). By checking the optimality condition of the sparsity-induced regularization problem (group lasso or MKL), the merit of active method is to iteratively include a violated kernel or group of features which mostly violates the optimality condition. Note, FGM differs from the active method in the following several aspects.

- The active-set algorithm is derived from the Lagrangian duality and the optimality conditions; while FGM starts from the AFS and AGS problem and then solves a semi-infinite QCQP problem via the cutting plane algorithm.

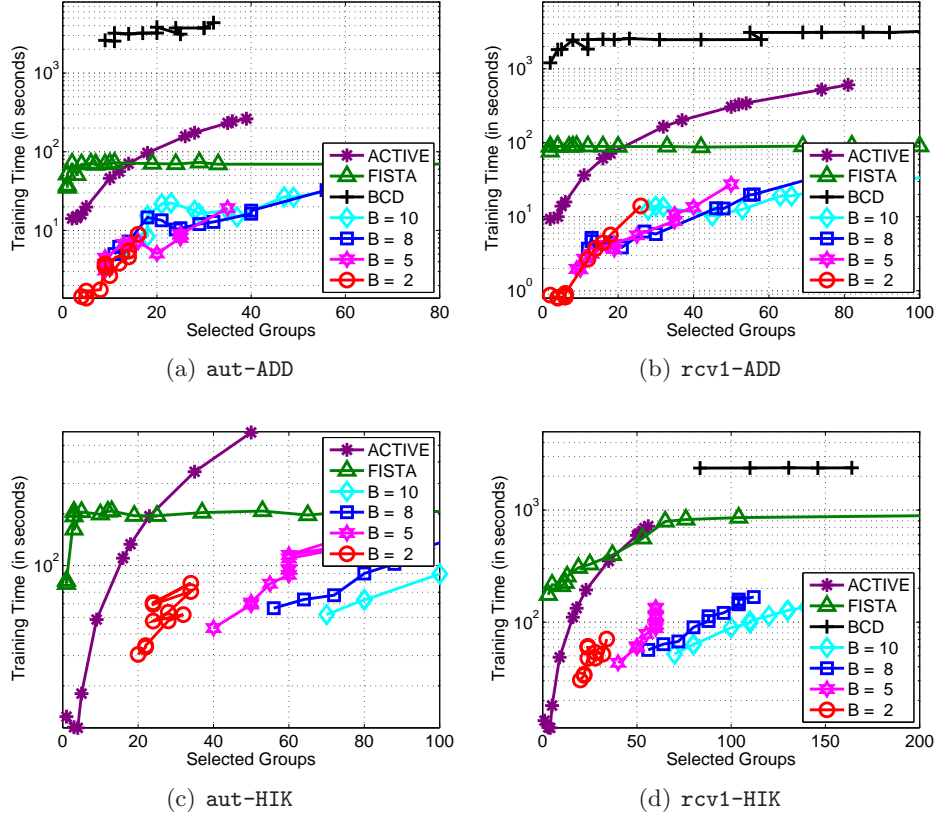


Figure 18: Training time on group selections. The groups are generated by either HIK or additive feature mappings. The result of BCD for aut-HIK cannot be available for the heavy computational cost.

- The active method starts from the standard group lasso and also solve a standard reduced group lasso problem (Bach, 2009; Roth and Fischer, 2008); while from our formulation, we need to solve a series of ℓ_1^2 -regularized problems, which are the primal forms of the MKL problem (29) with different number of kernels.
- At each iteration, the active methods only include one active kernel or group into the active constraint set. In this sense, the active method can be considered as a special case of FGM when $B = 1$. However, this strategy may not be efficient if the desired number of groups is relatively large. On the contrary, by extending FGM to group selection, it is natural to include $B \geq 1$ groups of features at each iteration of cutting plane algorithm. Such strategy can significantly improve the convergence speed over active set methods, which has been demonstrated in the experiments.
- Last but not the least, when doing group feature selections, the selected multiple groups in FGM will be formed as a new **super group** in FGM, which is quite different from the active method.

The feature generating scheme adapted in FGM also shares some merits as infinite kernel learning (IKL) in (Gehler and Nowozin, 2008), but FGM differs from IKL in several aspects. FGM is motivated from a feature scaling problem and to do feature selections. Secondly, FGM conducts the exact inference specifically designed for feature selections. Thirdly, in this paper, FGM solves the MKL subproblem in the primal. With this primal formulation, several techniques including APG update, cache techniques and warm starts strategy, have been proposed to further accelerate the training efficiency.

Finally, we discuss how FGM alleviates the bias problem in ℓ_1 -regularized methods. Taking the linear feature selection for example, the parameters B and C can be adjusted separately in FGM. For a fixed sparsity B , and a series of ℓ_1^2 -regularized problems will be solved in the feature generating scheme. When minimizing the ℓ_1^2 -regularized problem, because the included features in the worst-case analysis stage are the most informative features regarding the current loss ξ , one can use a relatively large trade-off parameter C to enforce that the empirical loss can be sufficiently minimized. In other words, a suitably large C can alleviate the bias brought by the variation of \mathbf{w} . On the contrary, in traditional ℓ_1 -regularized methods, if one wants to select a small number of informative features, one have to set a relatively smaller trade-off parameter C . However, in such situation, the empirical loss may not be sufficiently minimized. Accordingly, the bias problem may happen. To avoid this issue, we can do retraining after we obtain the selected features with a large C . However, as a smaller C may not make a sufficient minimization of the empirical loss, the selected features may not be very informative, either. These issues have been verified in the experiments. Interestingly, since the optimization scheme of the active set method is similar to FGM, with the same reasons discussed above, the active set method can also alleviate the bias problem, which can be observed from the experiments on group selections.

In summary, a **Feature Generating Machine** (FGM) is presented to learn the sparsity of features or feature groups, where a scaling vector $\mathbf{d} \in [0, 1]^m$ is introduced to measure the importance of features or groups. By imposing a ℓ_1 -norm constraint on \mathbf{d} , we transform the feature (group) scaling problem into a minimax optimization problem, which can be easily solved by an efficient and scalable cutting plane algorithm. The global convergence of the cutting plane algorithm has been provided under the exact solution to the worst-case analysis. In addition, with the separation of the complexity control and sparsity control, the bias problem can be alleviated. Finally, because FGM only needs to solve a small number of MKL problems, it is very suitable for solving large-scale and very high dimensional problems. Comprehensive experiments on both synthetic datasets and real-word datasets verify the good classification performance and training efficiency of the proposed strategies.

References

- F. R. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. In *Optimization for Machine Learning*. S. Sra, S. Nowozin, S. J. Wright., 2011.
- F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004.

- F. R. Bach. High-dimensional non-linear variable selection through hierarchical kernel learning. Technical report, 2009, 2009.
- A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.
- B. Blum, M. I. Jordan, David E. Kim, R. Das, P. Bradley, and D. Baker. Feature selection methods for improving protein structure prediction with rosetta. In *NIPS*, 2007.
- P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, 1998.
- A. B. Chan, N. Vasconcelos, and G. R. G. Lanckriet. Direct convex relaxations of sparse SVM. In *ICML*, 2007.
- Y.W. Chang, C.-J. Hsieh, . K.-W. Chang, M. Ringgaard, and C.-J. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *JMLR*, 11:1471–1490, 2010.
- O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.
- J. Chen and J. Ye. Training SVM with indefinite kernels. In *ICML*, 2008.
- A. Dasgupta, P. Drineas, and B. Harb. Feature selection methods for text classification. In *KDD*, 2007.
- R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification(second ed.)*. Hoboken, NJ: Wiley-Interscience, 2000.
- R. Fan, P. Chen, and C. Lin. Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889–1918, 2005. URL <http://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf>.
- M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE J. Sel. Topics Signal Proces*, 2007.
- G. M. Fung and O.L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28:185–202, 2004.
- P. Gehler and S. Nowozin. Infinite kernel learning. Technical report, MPI Technical Report 178, 2008.
- T. R. Golub, D. K. Slonim, and P. Tamayo. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 7:286–531, 1999.
- Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in svms. In *NIPS*, 2002.
- Q. Gu, Z. Li, and J. Han. Correlated multi-label feature selection. In *CIKM*, 2011a.
- Q. Gu, Z. Li, and J. Han. Generalized fisher score for feature selection. In *UAI*, 2011b.

- I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3: 1157–1182, 2003.
- I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.
- C. J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *ICML*, 2008.
- R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334, 2011.
- S. Ji, L. Yuan, Y. Li, Z. Zhou, S. Kumar, and J. Ye. Drosophila gene expression pattern annotation using sparse features and term-term interactions. In *KDD*, 2009.
- T. Joachims. Training linear SVMs in linear time. In *ACM KDD*, 2006.
- J. E. Kelley. The cutting plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.
- S. Kim and E. P. Xing. Tree-guided group lasso for multi-response regression with structured sparsity, with applications to eqtl mapping. *Annals of Applied Statistics*, Forthcoming, 2012.
- S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML 2010*, 2010.
- S.J. Kim and S. Boyd. A minimax theorem with applications to machine learning, signal processing, and finance. *SIAM Journal on Optimization*, 2008.
- M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. Lp-norm multiple kernel learning. *Journal of Machine Learning Research*, 12:953–997, 2011.
- G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.
- J. Liu and J. Ye. Efficient l1/lq norm regularization. Technical report, 2010.
- S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009.
- Andre F.T. Martins, Mario A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. Online multiple kernel learning for structured prediction. Technical report, 2010.
- A. Mutapcic and S. Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods & Software*, 24(3):381–406, 2009.
- A. Nedic and A. Ozdaglar. Subgradient methods for saddle-point problems. *Journal of Optimization Theory and Applications*, 142(1):205–228, 2009.
- A.Y. Ng. On feature selection: Learning with exponentially many irrelevant features as training examples. In *ICML*, 1998.

- E. Y. Pee and J. O. Royset. On solving large-scale finite minimax problems using exponential smoothing. *Journal of Optimization Theory and Applications*, online, 2010.
- Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. Technical report, 2010.
- R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97: 273–324, 1997.
- A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *ICML*, 2007.
- A. Rakotomamonjy, F. Bach, Y. Grandvalet, and S. Canu. SimpleMKL. *JMLR*, 9:2491–2521, 2008.
- A. Rakotomamonjy. Variable selection using svm-based criteria. *Journal of Machine Learning Research*, 3:1357–1370, 2003.
- M. Rastegari, C. Fang, and L. Torresani. Scalable object-class retrieval with approximate and top-ranking. In *ICCV*, 2011.
- V. Roth and B. Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *ICML*, 2008.
- S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large Scale Multiple Kernel Learning. *JMLR*, 7:1531–1565, 2006.
- M. Tan, L. Wang, and T. Ivor. Learning sparse svm for feature selection on very high dimensional datasets. In *ICML*, 2010.
- K.-C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. Technical report, 2009.
- M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML*, 2009.
- A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.
- J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *NIPS*, 2000.
- X. Wu, K. Yu, H. Wang, and W. Ding. Online streaming feature selection. In *ICML*, 2010.
- Z. Xu, R. Jin, Ye J., Michael R. Lyu, and King I. Non-monotonic feature selection. In *ICML*, 2009a.
- Z. Xu, R. Jin, I. King, and M.R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*. 2009b.
- Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, 2010.

- G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11, 2010.
- G.-X. Yuan, C.-H. Ho, and C.-J. Lin. An improved glmnet for l1-regularized logistic regression and support vector machines. In *KDD*, 2011.
- D. Zhang and W.S. Lee. Extracting keysubstringgroup features for text classification. In *KDD*, 2006.
- Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *ICML*, 2007.
- J. Zhu, S. Rossett, T. Hastie, and R. Tibshirani. ℓ_1 -norm support vector machines. In *NIPS*, 2003.

